EXHIBIT 1

EXHIBIT 1—CLAIM LISTING

Identifier	Claim Language	Status
	’969 Patent	
1[pre]	A portable device configured to communicate with a terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the portable device comprising:	**Invalid per Ingenico verdict**
1[a]	(a) an external communication interface configured to enable the transmission of communications between the portable device and the terminal;	**Invalid per Ingenico verdict**
1[b]	(b) a processor; and	**Invalid per Ingenico verdict**
1[c]	(c) a memory having executable program code stored thereon, including:	**Invalid per Ingenico verdict**
1[c][i]	(1) third program code which, when executed by the portable device processor, is configured to provide a communications node on the portable device to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and facilitate communications to the terminal and to a communications network node through the terminal network communication interface; and	**Invalid per Ingenico verdict**
1[c][ii]	(2) fourth program code which is configured to be executed by the portable device processor in response to a communication received by the portable device resulting from user interaction with the interactive user interface;	**Invalid per Ingenico verdict**
1[d]	wherein the portable device is configured to facilitate communications through the communication node on the terminal and the terminal network interface to a communications network	**Invalid per Ingenico verdict**

Identifier	Claim Language	Status
	node.	
2.	The portable device according to claim 1, wherein the fourth program code which, when executed by the portable device processor, is configured to cause a communication to be transmitted to the communication network node.	**Invalid per Ingenico verdict**
3.	The portable device according to claim 2, wherein the communication caused to be transmitted to the communication network node ***facilitates verification of the portable device***.	**Invalid per Ingenico verdict**
10.	The portable device according to claim 2, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates ***synchronizing content on the portable device with content on the communication network node database***.	**Currently asserted**

Identifier	Claim Language	Status
	’703 Patent	
104[pre]	A system implementing a terminal having a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to and from the terminal, the system comprising:	**Invalid per Ingenico verdict**
104[a]	(a) a communications network node; and	**Invalid per Ingenico verdict**
104[b]	(b) a portable device comprising an external communication interface for enabling the transmission of a plurality of communications between the portable device and the terminal, a processor, and a memory, wherein the memory has executable program code stored thereon, the portable device configured to:	**Invalid per Ingenico verdict**

Identifier	Claim Language	Status
104[b][i]	(1) cause the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component;	**Invalid per Ingenico verdict**
104[b][ii]	(2) execute third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal network communication interface;	**Invalid per Ingenico verdict**
104[b][iii]	(3) execute fourth program code stored on the portable device memory in response to a communication received by the portable device resulting from user interaction with the interactive user interface to cause a communication to be transmitted to a communications network node; and	**Invalid per Ingenico verdict**
104[b][iv]	(4) facilitate communications through the terminal network communication interface to a communications network node.	**Invalid per Ingenico verdict**
105.	The system according to claim 104, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate verification of the portable device.	**Invalid per Ingenico verdict**
114.	The system according to claim 104, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate ***synchronizing content on the portable device with content on the communications network node***.	**Currently asserted**
120.	120. The system according to claim 104, wherein the portable device is configured to execute the fourth program code to provide the terminal with data stored on the portable device to facilitate the terminal to transmit a communication to the communications network node.	**Invalid per Ingenico verdict**
124.	124. The system according to claim 120, wherein the data stored on the portable device memory comprises portable device identifier information.	**Invalid per Ingenico verdict**

EXHIBIT 2

US009774703B2

(12) **United States Patent**
McNulty

(10) **Patent No.:** **US 9,774,703 B2**
(45) **Date of Patent:** **Sep. 26, 2017**

(54) **APPARATUS, METHOD AND SYSTEM FOR A TUNNELING CLIENT ACCESS POINT**

(71) Applicant: **IOENGINE LLC**, Norwalk, CT (US)

(72) Inventor: **Scott McNulty**, Rowayton, CT (US)

(73) Assignee: **IOENGINE, LLC**, Norwalk, CT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 86 days.

(21) Appl. No.: **14/721,540**

(22) Filed: **May 26, 2015**

(65) **Prior Publication Data**

US 2015/0334208 A1 Nov. 19, 2015

Related U.S. Application Data

(63) Continuation of application No. 13/960,514, filed on Aug. 6, 2013, now Pat. No. 9,059,969, which is a continuation of application No. 12/950,321, filed on Nov. 19, 2010, now Pat. No. 8,539,047, which is a

(Continued)

(51) **Int. Cl.**

G06F 15/16	(2006.01)
G06F 15/177	(2006.01)
H04L 29/06	(2006.01)
H04L 29/08	(2006.01)
H04L 9/32	(2006.01)
G06F 13/00	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/42*** (2013.01); ***H04L 9/3226*** (2013.01); ***H04L 9/3247*** (2013.01); ***H04L 63/0272*** (2013.01); ***H04L 63/0428*** (2013.01); ***H04L 65/4069*** (2013.01); ***H04L 67/04*** (2013.01); ***H04L 67/141*** (2013.01); *H04L 2209/56* (2013.01); *H04L 2209/76* (2013.01); *H04L 2209/80* (2013.01)

(58) **Field of Classification Search**

CPC H04L 2209/76; H04L 2209/80; H04L 63/0272; H04L 63/0428; H04L 67/04; H04L 67/141; H04L 67/42; H04L 2209/56; H04L 65/4069; H04L 9/3226; H04L 9/3247

USPC 709/203, 217, 219, 220, 249; 711/115; 713/150

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,928,463	B1 *	8/2005	Tene	H04L 12/2856 370/356
6,986,030	B2	1/2006	Shmueli et al.	
7,051,157	B2	5/2006	James	

(Continued)

OTHER PUBLICATIONS

Microsoft Computer Dictionary, Fifth Edition, 2002, pp. 362, 437, 458, 565, and 572.

Defendant Interactive Media Corp. d/b/a Kanguru Solutions Initial Invalidity Contentions to Plaintiff IOENGINE, dated Jul. 14, 2015, 255 pages.

Imation's Initial Invalidity Contentions, dated Jul. 14, 2015, 248 pages.

(Continued)

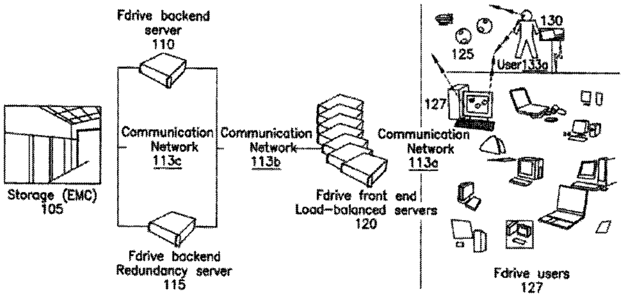
Primary Examiner — Alina N Boutah

(74) *Attorney, Agent, or Firm* — Locke Lord LLP

(57) **ABSTRACT**

The disclosure details the implementation of an apparatus, method, and system comprising a portable device configured to communicate with a terminal and a network server, and execute stored program code in response to user interaction with an interactive user interface. The portable device contains stored program code configured to render an interactive user interface on a terminal output component to enable the user the control processing activity on the portable device and access data and programs from the portable device and a network server.

129 Claims, 10 Drawing Sheets


Fdrive backend server 110
Storage (EMC) 105
Communication Network 113c
Communication Network 113b
Fdrive front end Load-balanced servers 120
Communication Network 113a
125
130
User 133a
127
Fdrive backend Redundancy server 115
Fdrive users 127

Related U.S. Application Data

continuation of application No. 10/807,731, filed on Mar. 23, 2004, now Pat. No. 7,861,006.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,546,340	B2 *	6/2009	Terasawa	G06F 1/1626 709/203
7,979,700	B2	7/2011	Elazar et al.	
8,595,488	B2	11/2013	Elazar et al.	
8,612,511	B2 *	12/2013	Friedrich	G05B 19/409 709/203
2002/0044663	A1	4/2002	King et al.	
2002/0046292	A1 *	4/2002	Tennison	H04L 12/5692 709/238
2002/0065872	A1	5/2002	Genske et al.	
2002/0080090	A1 *	6/2002	Borgstrom	G08C 17/02 345/1.1
2002/0194499	A1 *	12/2002	Audebert	H04L 63/0853 726/35
2003/0058274	A1 *	3/2003	Hill	H04L 63/0272 715/751
2004/0039932	A1	2/2004	Elazar et al.	
2005/0197859	A1 *	9/2005	Wilson	G06Q 50/22 705/2
2005/0198221	A1 *	9/2005	Manchester	H04L 41/0213 709/220
2006/0052085	A1 *	3/2006	Gregrio Rodriguez	H04L 12/2859 455/411
2007/0038870	A1 *	2/2007	Ciesinger	H04L 63/0428 713/193

OTHER PUBLICATIONS

Scott Spanbauer, “Mighty Mini Media”, www.pcworld.com, May 2002, pp. 14-17.

Miranda Instant Messenger, https://web.archive.org/web/20031228092924/http:/www.miranda-im.org, Copyright 2000-2003 Miranda IM, 2 pages.

Miranda Instant Messenger (2), “About Miranda IM”, https://web.archive.org/web/20031228092924/http:/www.miranda-im.org, Copyright 2000-2003 Miranda IM, 2 pages.

Miranda Instant Messenger (3), “Screenshots”, https://web.archive.org/web/20031228092924/http:/www.miranda-im.org, Copyright 2000-2003 Miranda IM, 2 pages.

Jon L. Jacoby, Welcome to M-Systems DiskOnKey Site, https://web/archive.org/web/20021202082914/http://www.diskonkey.com/prod_dok.asp, 2 pages.

Jon L. Jacoby, Welcome to M-Systems DiskOnKey Site, “Product & Solutions”, https://web/archive.org/web/20021202082914/http://www.diskonkey.com/prod_dok.asp, 1 page.

Jon L. Jacoby, M-Systems Flash Disk Pioneers, “Using MyKey”, Copyright 2003 M-Systems Flash Disk Pioneers, Ltd., 23 pages.

Expert Report by Vijay Madiselli, Ph.D., *Ioengine, LLC* v. *Interactive Media Corp.*, C.A. No. 14-1571 (D.Del.) and *Ioengine, LLC* v. *Imation Corp.*, C.A. No. 14-1572 (D.Del.) Jul. 1, 2016 (141 pages).

Rebuttal Expert Report of Dr. Kevin Butler Regarding the Validity of U.S. Pat. No. 8,539,047, *Ioengine, LLC* v. *Interactive Media Corp.*, C.A. No. 14-1571 (D.Del.) and *Ioengine, LLC* v. *Imation Corp.*, C.A. No. 14-1572 (D.Del.) Jul. 22, 2016 (78 pages).

* cited by examiner

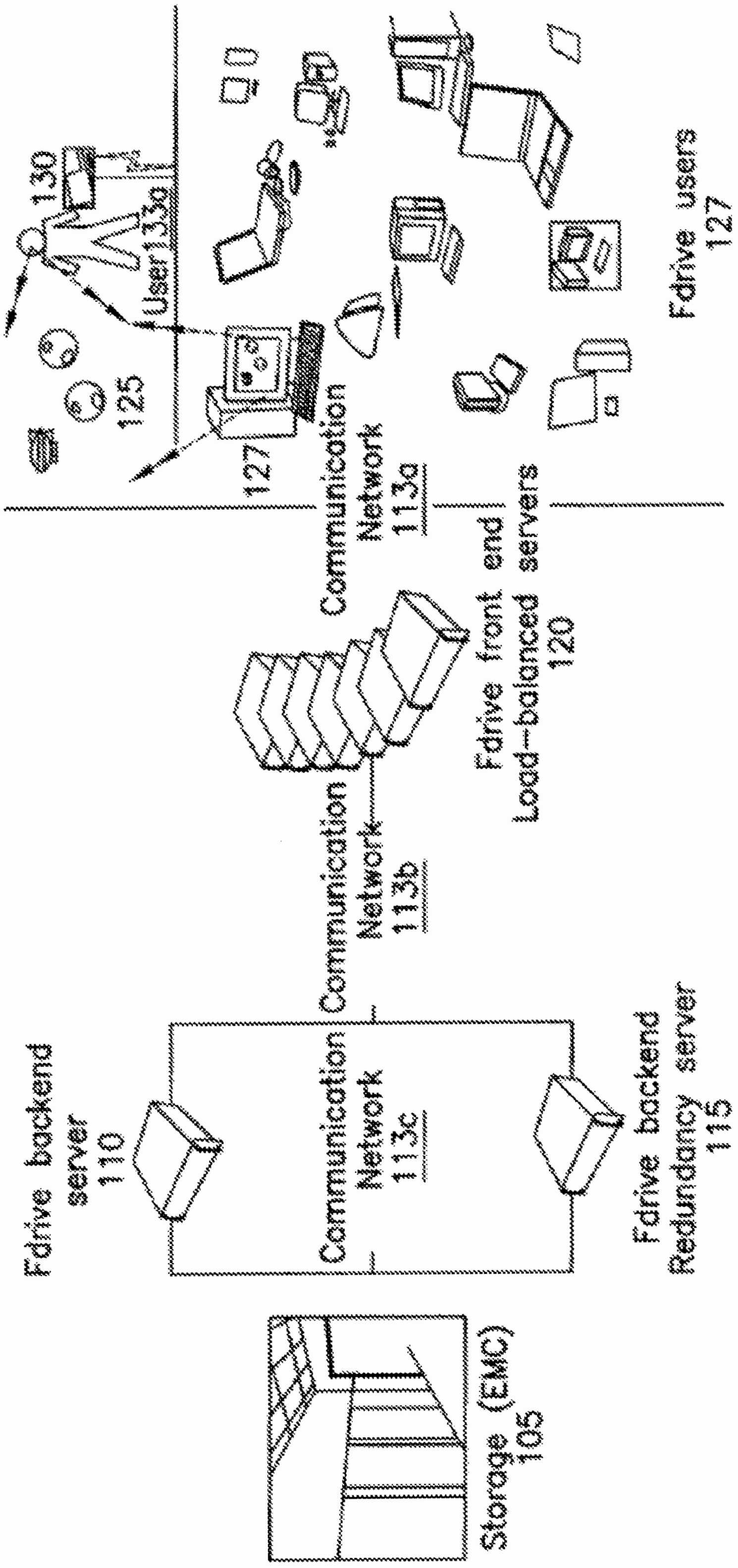

Storage (EMC)
105
Fdrive backend
server
110
Communication
Network
113c
Fdrive backend
Redundancy server
115
Communication
Network
113b
Fdrive front end
Load-balanced servers
120
Communication
Network
113a
127
125
User 133a
130
Fdrive users
127

Fig. 1

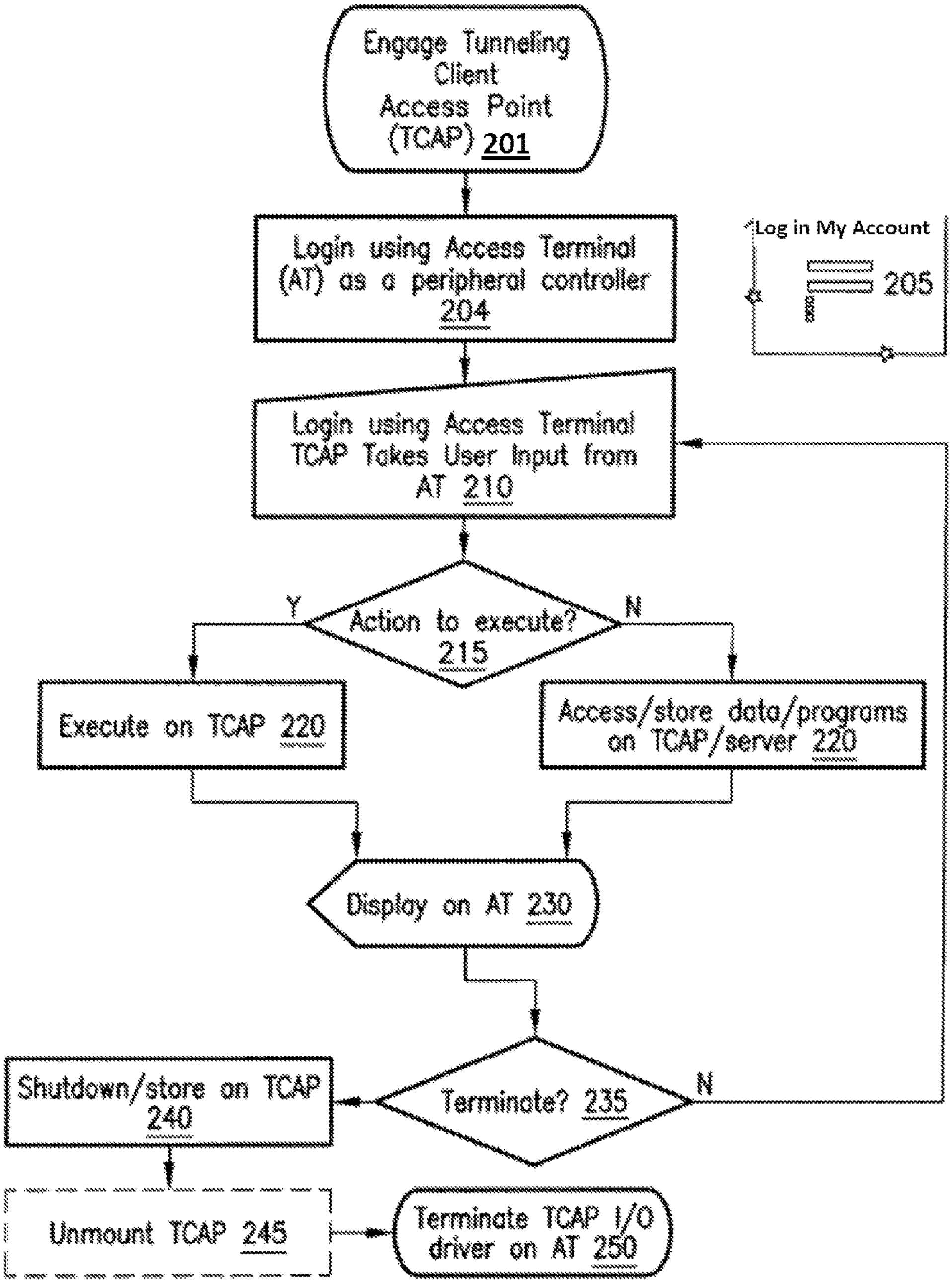

Engage Tunneling Client Access Point (TCAP) 201
Login using Access Terminal (AT) as a peripheral controller 204
Log in My Account
205
Login using Access Terminal TCAP Takes User Input from AT 210
Y
Action to execute? 215
N
Execute on TCAP 220
Access/store data/programs on TCAP/server 220
Display on AT 230
Shutdown/store on TCAP 240
Terminate? 235
N
Unmount TCAP 245
Terminate TCAP I/O driver on AT 250

Fig. 2

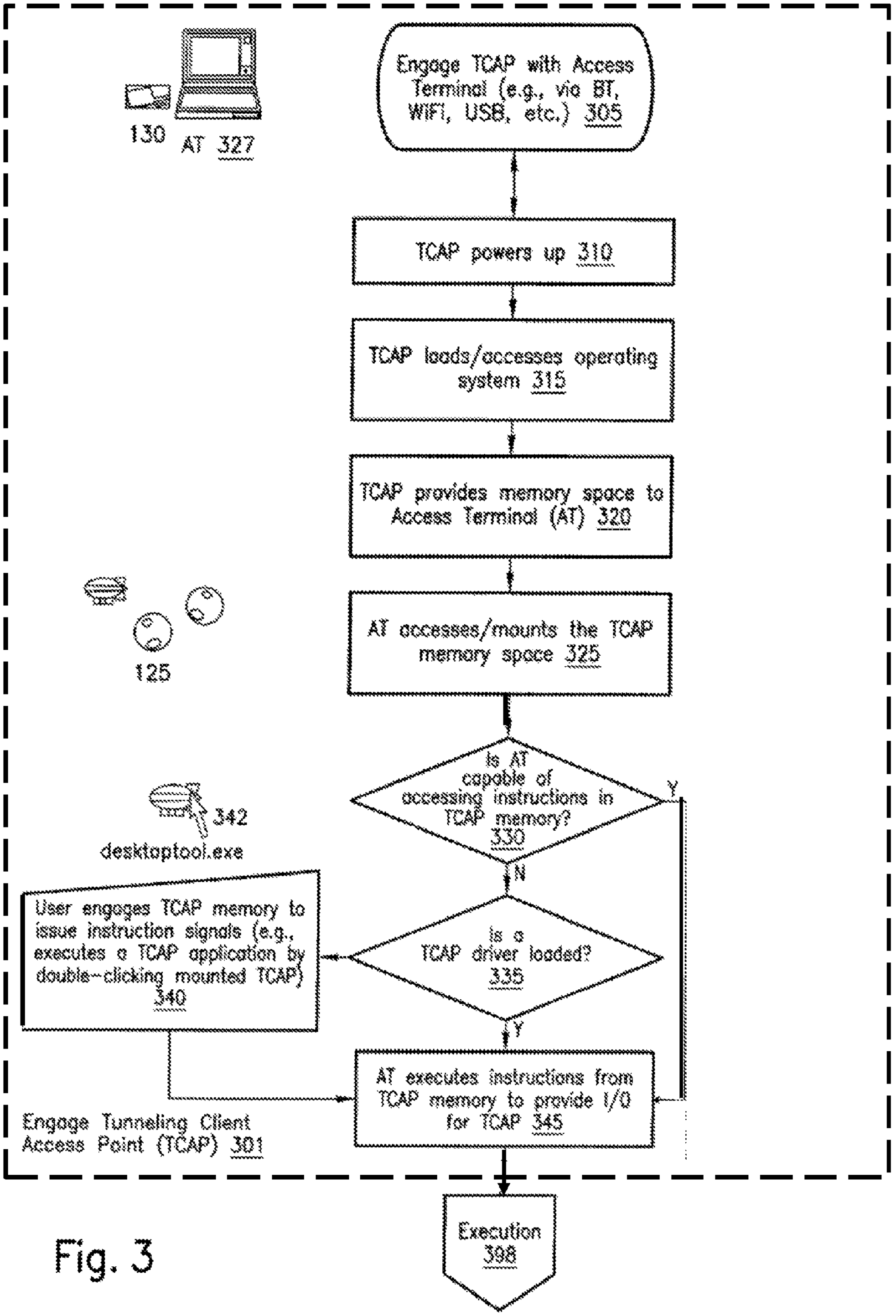

130
AT 327
Engage TCAP with Access Terminal (e.g., via BT, WiFi, USB, etc.) 305
TCAP powers up 310
TCAP loads/accesses operating system 315
TCAP provides memory space to Access Terminal (AT) 320
125
AT accesses/mounts the TCAP memory space 325
342
desktoptool.exe
Is AT capable of accessing instructions in TCAP memory? 330
Y
N
User engages TCAP memory to issue instruction signals (e.g., executes a TCAP application by double-clicking mounted TCAP) 340
Is a TCAP driver loaded? 335
Y
AT executes instructions from TCAP memory to provide I/O for TCAP 345
Engage Tunneling Client Access Point (TCAP) 301
Execution 398

Fig. 3

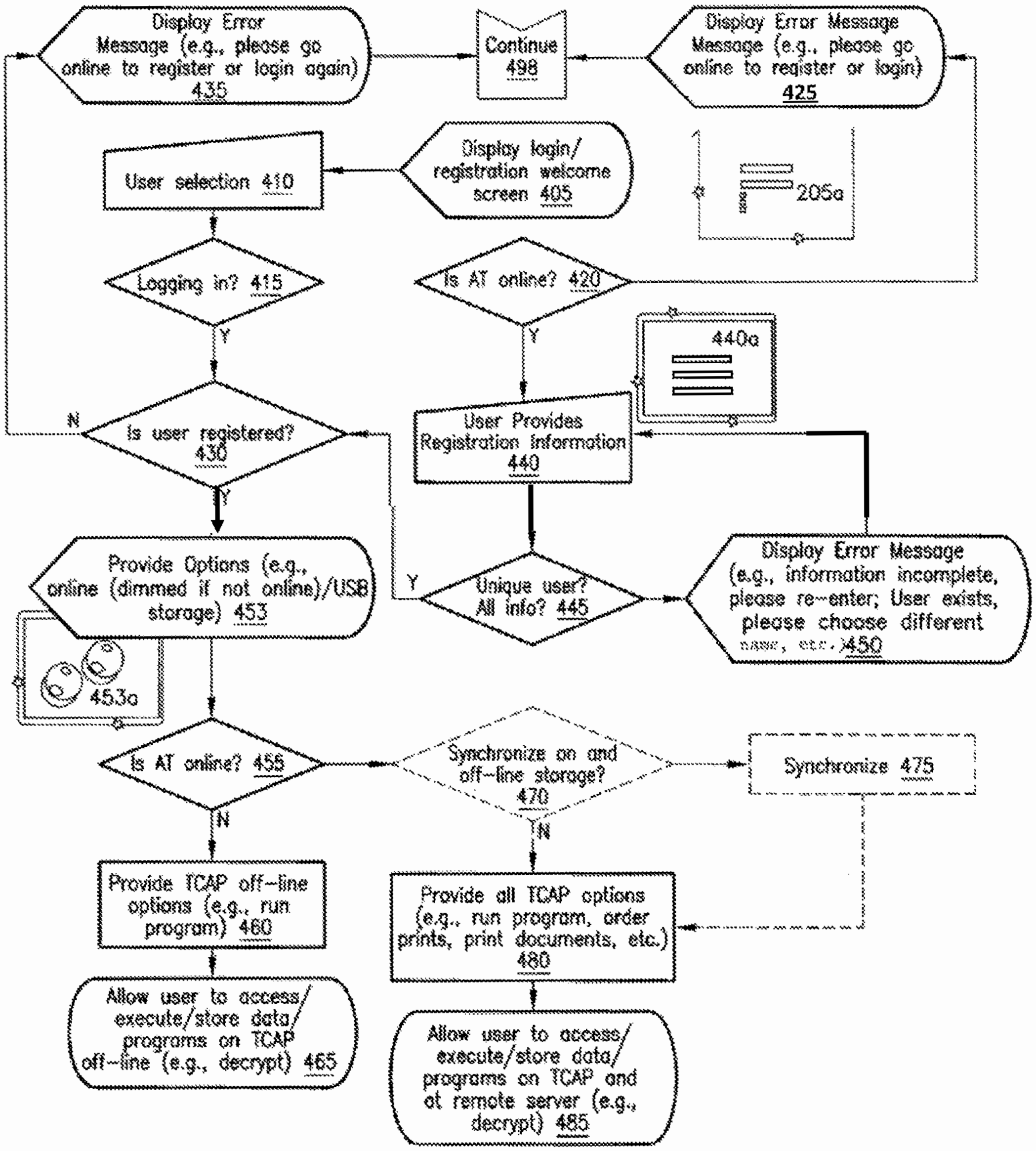

Display Error Message (e.g., please go online to register or login again) 435
Continue 498
Display Error Message (e.g., please go online to register or login) 425
User selection 410
Display login/ registration welcome screen 405
205a
Logging in? 415
Is AT online? 420
Y
Y
440a
N
Is user registered? 430
User Provides Registration information 440
Y
Provide Options (e.g., online (dimmed if not online)/USB storage) 453
Y
Unique user? All info? 445
Display Error Message (e.g., information incomplete, please re-enter; User exists, please choose different name, etc.) 450
453a
Is AT online? 455
Synchronize on and off-line storage? 470
Synchronize 475
N
N
Provide TCAP off-line options (e.g., run program) 460
Provide all TCAP options (e.g., run program, order prints, print documents, etc.) 480
Allow user to access/ execute/store data/ programs on TCAP off-line (e.g., decrypt) 465
Allow user to access/ execute/store data/ programs on TCAP and at remote server (e.g., decrypt) 485

Fig. 4

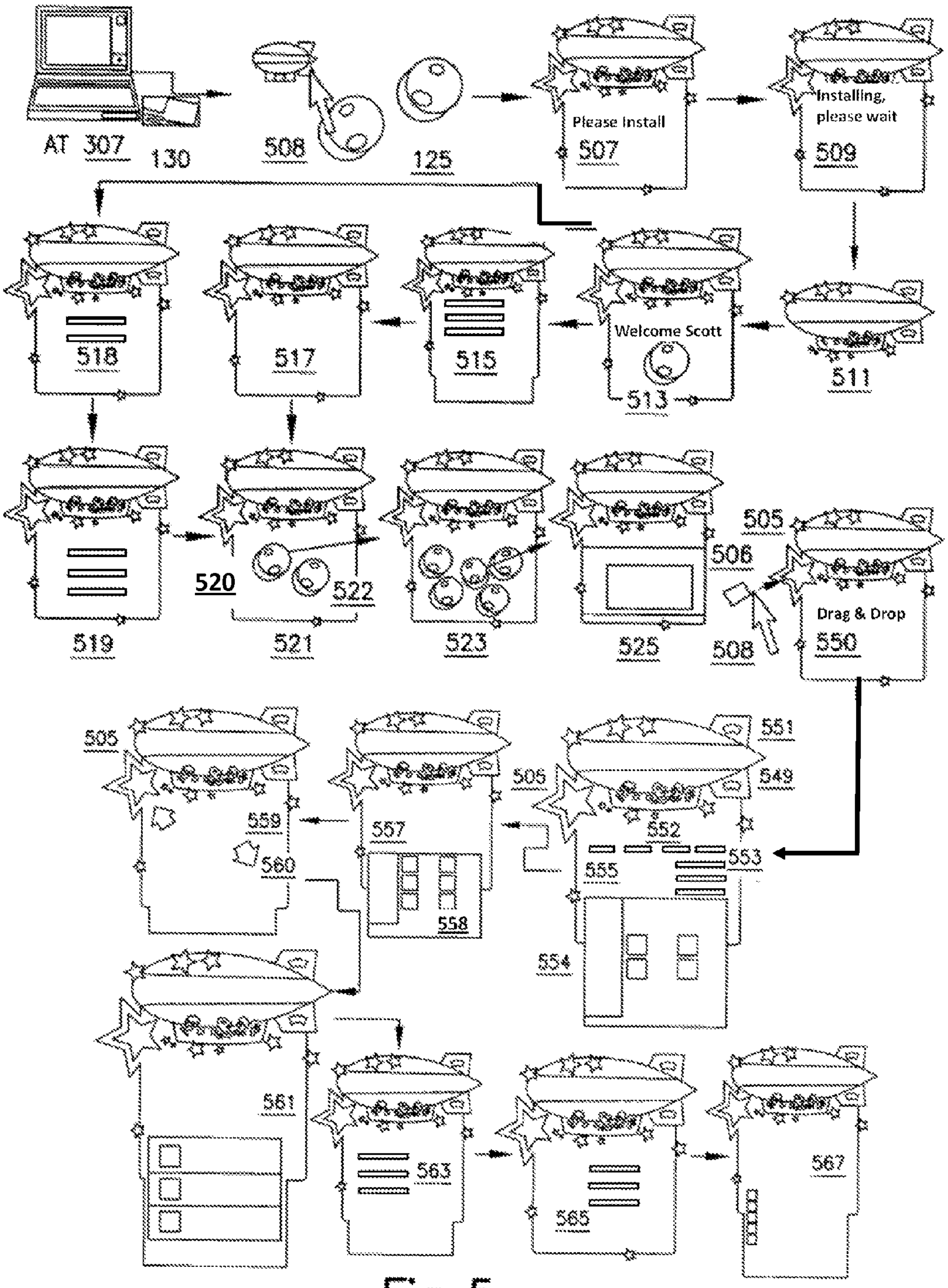

AT 307
130
508
125
Please Install
507
Installing, please wait
509
518
517
515
Welcome Scott
513
511
519
520
521
522
523
525
505
506
508
Drag & Drop
550
505
559
560
557
558
505
551
549
552
553
555
554
561
563
565
567

Fig. 5

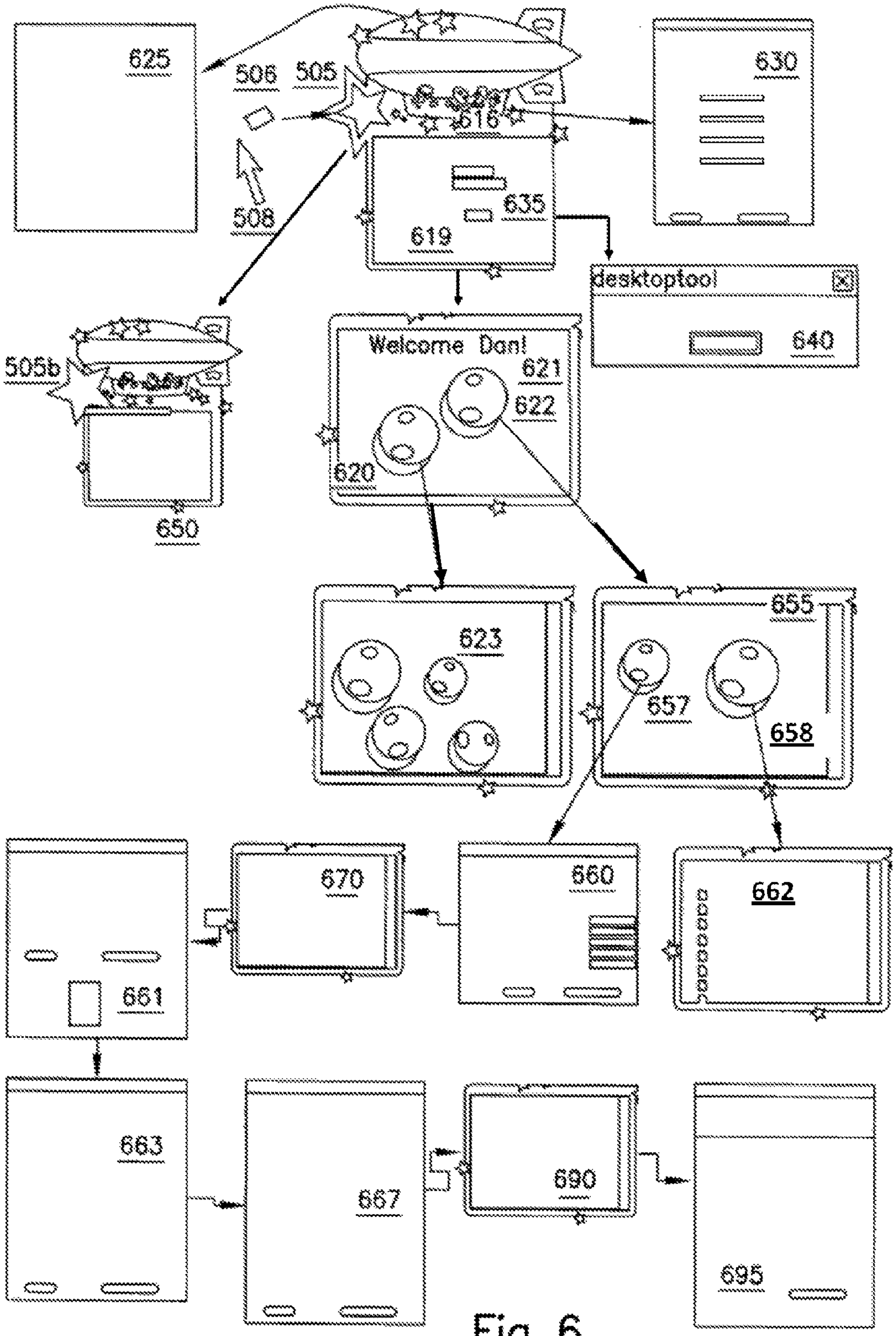

625
506
505
616
630
508
635
619
desktoptool
640
505b
Welcome Dan!
621
622
620
650
623
655
657
658
670
660
662
661
663
667
690
695

Fig. 6

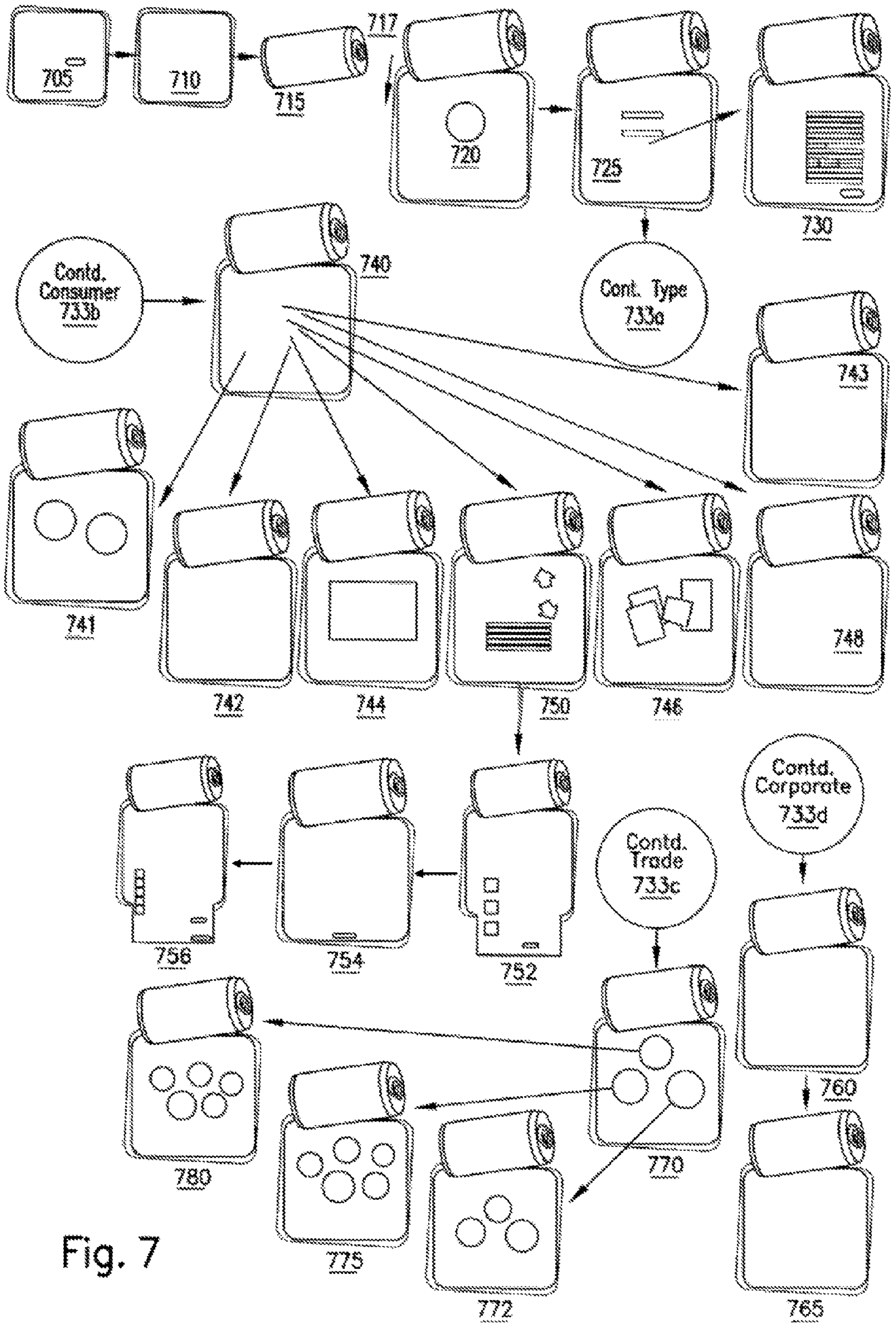

705
710
715
717
720
725
730
Contd. Consumer 733b
740
Cont. Type 733a
743
741
742
744
750
746
748
756
754
752
Contd. Trade 733c
Contd. Corporate 733d
760
780
775
772
770
765

Fig. 7

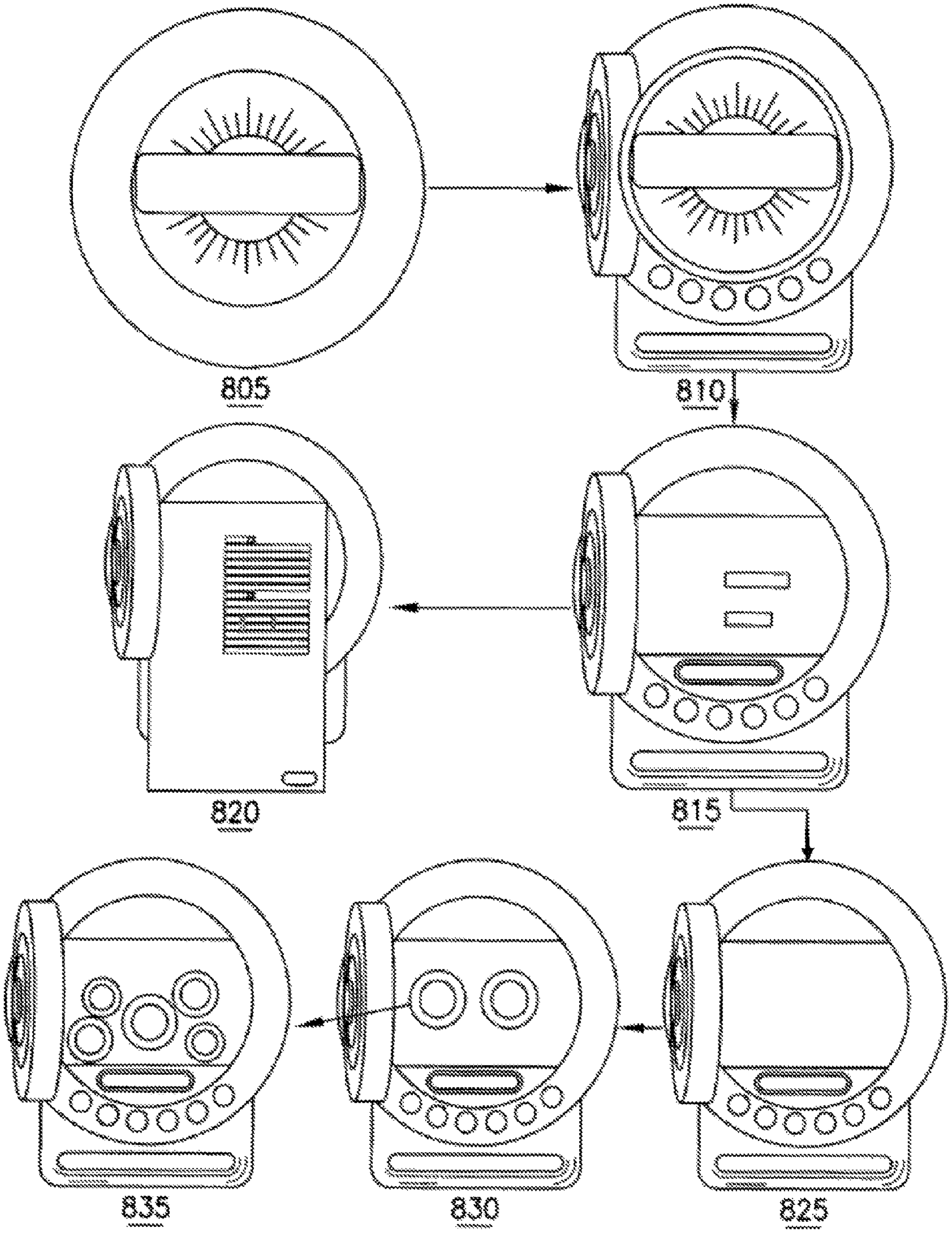


Fig. 8

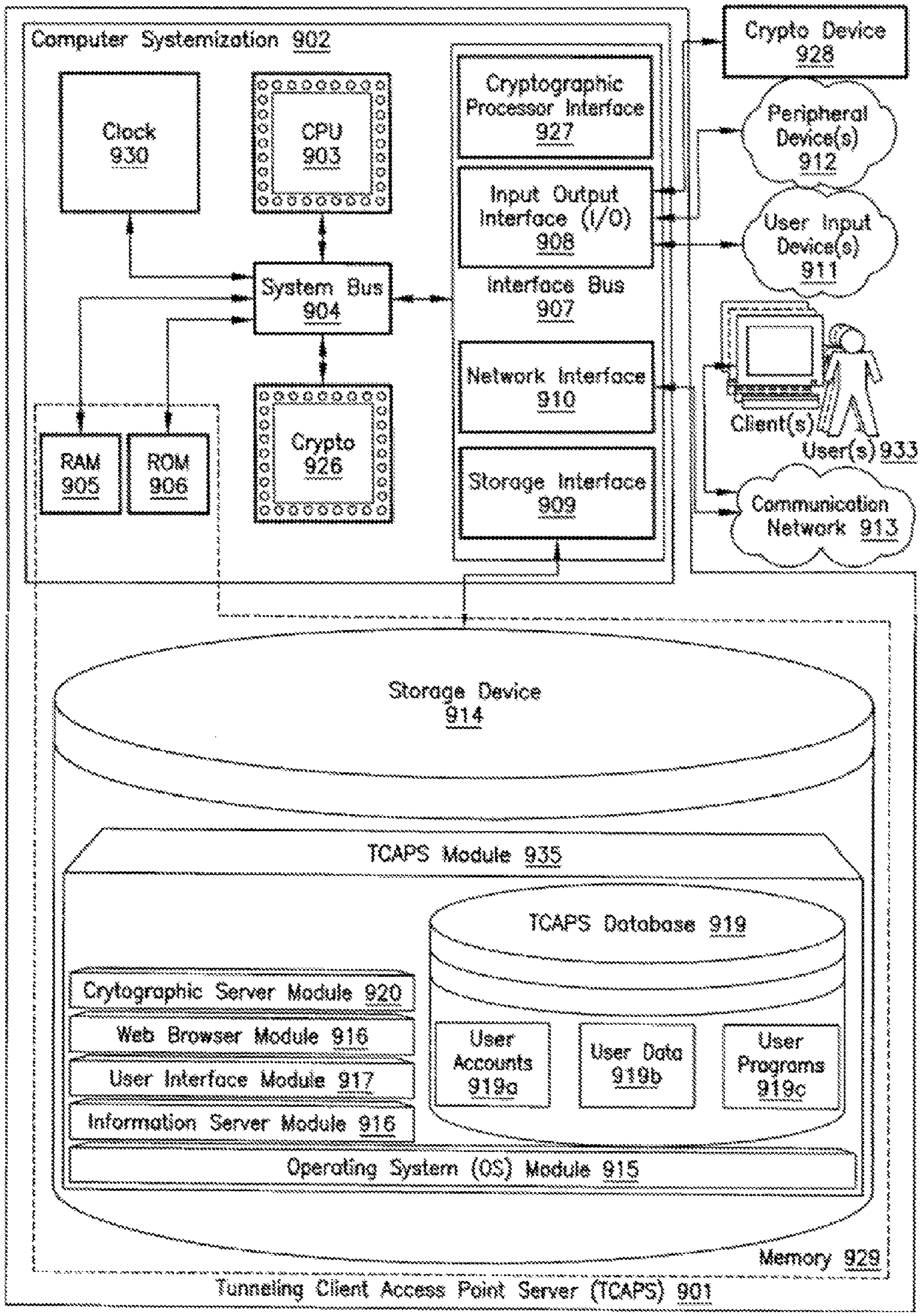
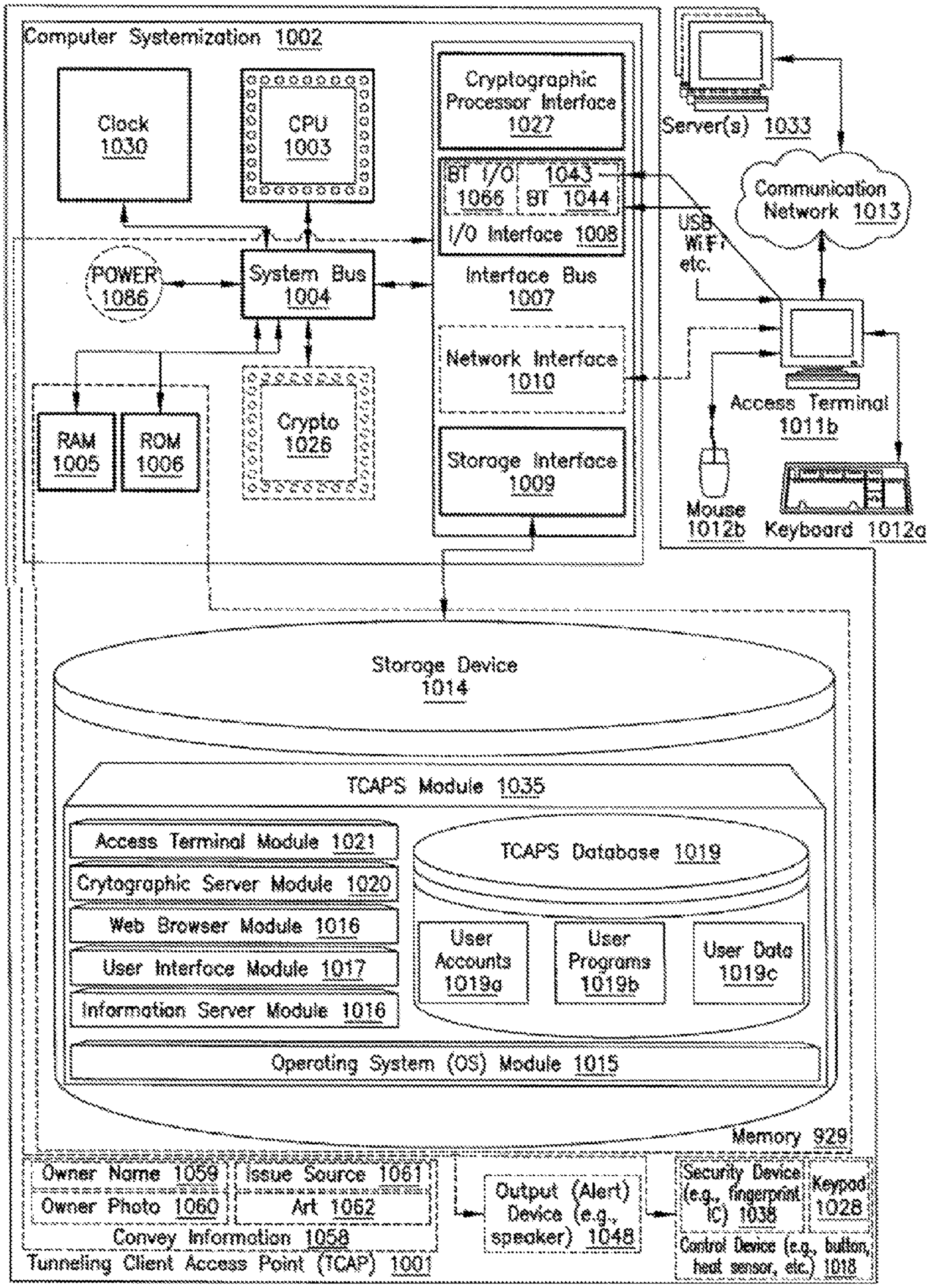

Computer Systemization 902
Clock 930
CPU 903
Cryptographic Processor Interface 927
Input Output Interface (I/O) 908
Interface Bus 907
System Bus 904
Network Interface 910
Storage Interface 909
RAM 905
ROM 906
Crypto 926
Crypto Device 928
Peripheral Device(s) 912
User Input Device(s) 911
Client(s)
User(s) 933
Communication Network 913
Storage Device 914
TCAPS Module 935
TCAPS Database 919
Crytographic Server Module 920
Web Browser Module 916
User Interface Module 917
Information Server Module 916
User Accounts 919a
User Data 919b
User Programs 919c
Operating System (OS) Module 915
Memory 929
Tunneling Client Access Point Server (TCAPS) 901

Fig. 9



Computer Systemization 1002
Clock 1030
CPU 1003
Cryptographic Processor Interface 1027
BT I/O 1066
1043
BT 1044
I/O Interface 1008
Server(s) 1033
Communication Network 1013
USB WiFi etc.
POWER 1086
System Bus 1004
Interface Bus 1007
Access Terminal 1011b
Network Interface 1010
RAM 1005
ROM 1006
Crypto 1026
Storage Interface 1009
Mouse 1012b
Keyboard 1012a
Storage Device 1014
TCAPS Module 1035
Access Terminal Module 1021
Crytographic Server Module 1020
Web Browser Module 1016
User Interface Module 1017
Information Server Module 1016
TCAPS Database 1019
User Accounts 1019a
User Programs 1019b
User Data 1019c
Operating System (OS) Module 1015
Memory 929
Owner Name 1059
Owner Photo 1060
Issue Source 1061
Art 1062
Convey Information 1058
Tunneling Client Access Point (TCAP) 1001
Output (Alert) Device (e.g., speaker) 1048
Security Device (e.g., fingerprint IC) 1038
Keypad 1028
Control Device (e.g., button, heat sensor, etc.) 1018

Fig.10

APPARATUS, METHOD AND SYSTEM FOR A TUNNELING CLIENT ACCESS POINT

This application is a continuation of U.S. application Ser. No. 13/960,514, filed Aug. 6, 2013, which is a continuation of U.S. application Ser. No. 12/950,321, filed Nov. 19, 2010, now U.S. Pat. No. 8,539,047, which is a continuation of U.S. application Ser. No. 10/807,731, filed on Mar. 23, 2003, now U.S. Pat. No. 7,861,006.

FIELD

The present invention is directed generally to an apparatus, method, and system of accessing data, and more particularly, to an apparatus, method and system to transmit and process data comprising a portable device in communication with a terminal and a communications network comprising a plurality of communications network nodes.

BACKGROUND

Portable Computing and Storage

Computing devices have been becoming smaller over time. Currently, some of the smallest computing devices are in the form of personal digital assistants (PDAs). Such devices usually come with a touch screen, an input stylus and/or mini keyboard, and battery source. These devices, typically, have storage capacities around 64 MB. Examples of these devices include Palm's Palm Pilot.

Information Technology Systems

Typically, users, which may be people and/or other systems, engage information technology systems (e.g., commonly computers) to facilitate information processing. In turn, computers employ processors to process information; such processors are often referred to as central processing units (CPU). A common form of processor is referred to as a microprocessor. A computer operating system, which, typically, is software executed by CPU on a computer, enables and facilitates users to access and operate computer information technology and resources. Common resources employed in information technology systems include: input and output mechanisms through which data may pass into and out of a computer; memory storage into which data may be saved; and processors by which information may be processed. Often information technology systems are used to collect data for later retrieval, analysis, and manipulation, commonly, which is facilitated through database software. Information technology systems provide interfaces that allow users to access and operate various system components.

User Interface

The function of computer interfaces in some respects is similar to automobile operation interfaces. Automobile operation interface elements such as steering wheels, gearshifts, and speedometers facilitate the access, operation, and display of automobile resources, functionality, and status. Computer interaction interface elements such as check boxes, cursors, menus, scrollers, and windows (collectively and commonly referred to as widgets) similarly facilitate the access, operation, and display of data and computer hardware and operating system resources, functionality, and status. Operation interfaces are commonly called user interfaces. Graphical user interfaces (GUIs) such as the Apple Macintosh Operating System's Aqua, Microsoft's Windows XP, or Unix's X-Windows provide a baseline and means of accessing and displaying information, graphically, to users.

Networks

Networks are commonly thought to comprise of the interconnection and interoperation of clients, servers, and intermediary nodes in a graph topology. It should be noted that the term "server" as used herein refers generally to a computer, other device, software, or combination thereof that processes and responds to the requests of remote users across a communications network. Servers serve their information to requesting "clients." The term "client" as used herein refers generally to a computer, other device, software, or combination thereof that is capable of processing and making requests and obtaining and processing any responses from servers across a communications network. A computer, other device, software, or combination thereof that facilitates, processes information and requests, and/or furthers the passage of information from a source user to a destination user is commonly referred to as a "node." Networks are generally thought to facilitate the transfer of information from source points to destinations. A node specifically tasked with furthering the passage of information from a source to a destination is commonly called a "router." There are many forms of networks such as Local Area Networks (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc. For example, the Internet is generally accepted as being an interconnection of a multitude of networks whereby remote clients and servers may access and interoperate with one another.

SUMMARY

Although all of the aforementioned portable computing systems exist, no effective solution to securely access, execute, and process data is available in an extremely compact form. Currently, PDAs, which are considered among the smallest portable computing solution, are bulky, provide uncomfortably small user interfaces, and require too much power to maintain their data. Current PDA designs are complicated and cost a lot because they require great processing resources to provide custom user interfaces and operating systems. Further, current PDAs are generally limited in the amount of data they can store or access. No solution exists that allows users to employ traditional large user interfaces they are already comfortable with, provides greater portability, provides greater memory footprints, draws less power, and provides security for data on the device. As such, the disclosed tunneling client access point (TCAP) is very easy to use; at most it requires the user to simply plug the device into any existing and available desktop or laptop computer, through which, the TCAP can make use of a traditional user interface and input/output (I/O) peripherals, while the TCAP itself, otherwise, provides storage, execution, and/or processing resources. Thus, the TCAP requires no power source to maintain its data and allows for a highly portable "thumb" footprint. Also, by providing the equivalent of a plug-n-play virtual private network (VPN), the TCAP provides certain kinds of accessing of remote data in an easy and secure manner that was unavailable in the prior art.

In accordance with certain aspects of the disclosure, the above-identified problems of limited computing devices are overcome and a technical advance is achieved in the art of portable computing and data access. An exemplary tunneling client access point (TCAP) includes a method to dispose a portable storage device in communication with a terminal. The method includes providing the memory for access on the terminal, executing processing instructions from the memory on the terminal to access the terminal, communicating through a conduit, and processing the processing instructions.

In accordance with another embodiment, a portable tunneling storage processor is disclosed. The apparatus has a memory and a processor disposed in communication with the memory, and configured to issue a plurality of processing instructions stored in the memory. Also, the apparatus has a conduit for external communications disposed in communication with the processor, configured to issue a plurality of communication instructions as provided by the processor, configured to issue the communication instructions as signals to engage in communications with other devices having compatible conduits, and configured to receive signals issued from the compatible conduits.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate various non-limiting, example, inventive aspects in accordance with the present disclosure:

FIG. **1** is of a flow diagram illustrating embodiments of a tunneling client access point (TCAP);

FIG. **2** is of a flow diagram illustrating embodiments of a system of tunneling client access point and access terminal interaction;

FIG. **3** is of a flow diagram illustrating embodiments of engaging the tunneling client access point to an access terminal interaction;

FIG. **4** is of a flow diagram illustrating embodiments of accessing the tunneling client access point and server through an access terminal;

FIGS. **5-8** is of a flow diagram illustrating embodiments of facilities, programs, and/or services that the tunneling client access point and server may provide to the user as accessed through an access terminal;

FIG. **9** is of a block diagram illustrating embodiments of a tunneling client access point server controller;

FIG. **10** is of a block diagram illustrating embodiments of a tunneling client access point controller;

The leading number of each reference number within the drawings indicates the first figure in which that reference number is introduced. As such, reference number **101** is first introduced in FIG. **1**. Reference number **201** is first introduced in FIG. **2**, etc.

DETAILED DESCRIPTION

Topology

FIG. **1** illustrates embodiments for a topology between a tunneling client access point (TCAP) (see FIG. **10** for more details on the TCAP) and TCAP server (TCAPS) (see FIG. **9** for more details on the TCAPS). In this embodiment, a user **133***a* may plug-in a TCAP into any number of access terminals **127** located anywhere. Access terminals (ATs) may be any number of computing devices such as servers, workstations, desktop computers, laptops, portable digital assistants (PDAs), and/or the like. The type of AT used is not important other than the device should provide a compatible mechanism of engagement to the TCAP **130** and provide an operating environment for the user to engage the TCAP through the AT. In one embodiment, the TCAP provides a universal serial bus (USB) connector through which it may plug into an AT. In other embodiment, the TCAP may employ Bluetooth, WiFi and/or other wireless connectivity protocols to connect with ATs that are also so equipped. In one embodiment, the AT provides Java and/or Windows runtime environments, which allows the TCAP to interact with the input/output mechanisms of the AT. See FIG. **9** for more details and embodiments on the types of connections that may be employed by the TCAP. Once the TCAP has engaged with an AT, it can provide the user with access to its storage and processing facilities.

If the AT is connected to a communication network **113**, the TCAP may then communicate beyond the AT. In one embodiment, the TCAP can provide extended storage and/or processing resources by engaging servers **110**, **115**, **120**, which have access to and can provide extended storage **105** to the TCAP through the AT. In one embodiment, a single server and storage device may provide such TCAP server support. In another embodiment, server support is provided over a communications network, e.g., the Internet, by an array of front-end load-balancing servers **120**. These servers can provide access to storage facilities within the servers or to remote storage **105** across a communications network **113***b, c* (e.g., a local area network (LAN)). In such an embodiment, a backend server **110** may offload the front-end server with regard to data access to provide greater throughput. For purposes of load balancing and/or redundancy, a backup server **115** may be similarly situated to provide for access and backup in an efficient manner. In such an embodiment, the back-end servers may be connected to the front-end servers through a communications network **113***b* (e.g., wide area network (WAN)). The backend servers **110**, **115** may be connected to the remote storage **105** through a communications network **113***c* as well (e.g., a high speed LAN, fiber-channel, and/or the like).

Thus, to the user **133***a*, the contents of the TCAP **130** appear on the AT as being contained on the TCAP **125** even though much of the contents may actually reside on the servers **115**, **120** and/or the servers' storage facilities **105**. In these ways, the TCAP "tunnels" data through an AT. The data may be provided through the AT's I/O for the user to observe without it actually residing on the AT. Also, the TCAP may tunnel data through an AT across a communications network to access remote servers without requiring its own more complicated set of peripherals and I/O.

TCAP and AT Interaction

FIG. **2** illustrates embodiments for a system of tunneling client access point (TCAP) (see FIG. **10** for more details on the TCAP) and access terminal interaction. FIG. **2** provides an overview for TCAP and AT interaction and subsequent figures will provide greater detail on elements of the interaction. In this embodiment, a user engages the TCAP **201**. For example, the user may plug the TCAP into an AT via the AT's USB port. Thereafter the user is presented with a login prompt **205** on the AT's display mechanism, e.g., on a video monitor. After a user successfully logs in (for example by providing a user name and password) **204**, the TCAP can then accept user inputs from the AT and its peripherals (the TCAP can then also provide output to the user via the AT's peripherals).

The user may employ the AT's input peripherals as user input devices that control actions on the TCAP. Depending on the user's actions **215**, the TCAP can be used by the AT as a storage device from which it can access and store data and programs **225**. For example, if the user takes the action of opening a file from the TCAP's memory, e.g., by double clicking on an icon when the TCAP is mounted as a USB drive on the AT, then the AT may treat the TCAP as a memory device and retrieve information from the TCAP **225**. If the user's action **215** is one that is directed at executing on the TCAP **215**, then the AT will not be involved in any execution. For example, if the user drops an icon representing a graphics file onto a drag-and-drop location visually representing the TCAP, then the file may be copied to the TCAP where it will process and spool the file for sending the graphics file to be printed at a remote location. In such a case, all of the requirements to process and spool the file are handled by the TCAP's processor and the AT would only be used as a mechanism for user input and output and as a conduit through which the TCAP may send files.

Regardless of if there is an action **215** to execute on the TCAP **220** or to access or store data on the TCAP **225**, the AT is used to display the status of any actions **230**. At any time the user may select to terminate TCAP related facilities executing either on the AT, a backend server, on the TCAP itself, and/or the like **235**. In one embodiment, the user may select a quit option that is displayed on the AT's screen. In another embodiment, the user may simply disengage the TCAP from the AT by severing the connection (e.g., turning power off, physically pulling the device off the AT, turning off wireless transmissions, and/or the like). It should be noted that such abrupt severing may result in the loss of data, file corruption, etc. if the TCAP has not saved data that is on the AT or on some remote server, however, if the TCAP is employing flash like memory, its contents should remain intact.

If there is no instruction signal to terminate the TCAP **235**, execution will continue and the TCAP will continue to take and look for input from the user. Of course if the TCAP has been set to perform certain actions, those actions will continue to execute, and the TCAP may respond to remote servers when it is communicating with them through the AT. When the user issues a terminate signal **235**, then the TCAP will shut down by saving any data to the TCAP that is in the AT's memory and then terminating any programs executing on both the AT and TCAP that were executed by and/or from the TCAP **240**. If no activities are taking place on the TCAP and all the data is written back to the TCAP **240**, then the TCAP may optionally unmount itself from the AT's file-system **245**. At this point, if there is a TCAP I/O driver executing on the AT, that driver may be terminated as triggered by the absence of the TCAP at a mount point **250**. After the TCAP is unmounted and/or the TCAP I/O driver is terminated, it is safe to disengage the TCAP from the AT.

TCAP and AT Interaction

FIG. **3** illustrates embodiments engaging the tunneling client access point to an access terminal interaction. Examples of engaging the TCAP **301** with an AT were discussed above in FIG. **1** **127**, **130**, **133***a* and FIG. **2** **201**. In one embodiment, the TCAP **130** is engaged with an access terminal **327**, **305**. As mentioned in FIG. **1**, the TCAP is capable of engaging with ATs using a number of mechanisms. In one embodiment, the TCAP has a USB connector for plugging into an AT, which acts as a conduit for power and data transfer. In another embodiment, the TCAP may use Bluetooth to establish a wireless connection with a number of ATs. In another embodiment, the TCAP may employ WiFi. In yet another embodiment, the TCAP may employ multiple communications mechanisms. It should be noted, with some wireless mechanisms like Bluetooth and WiFi, simply coming into proximity with an AT that is configured for such wireless communication may result in the TCAP engaging with and establish a communications link with the AT. In one embodiment, the TCAP has a "connect" button that will allow such otherwise automatically engaging interactions take place only if the "connect" button is engaged by a user. Such an implementation may provide greater security for users (see FIG. **10** for more details on the TCAP).

After being engaged **305**, the TCAP will then power on. In an embodiment requiring a direct connection, e.g., USB, simply plugging the TCAP into the AT provides power. In a wireless embodiment, the TCAP may be on in a lower powered state or otherwise turned on by engaging the connect button as discussed above. In such an embodiment, the TCAP can employ various on-board power sources (see FIG. **10** for more details on the TCAP). The TCAP then may load its own operating system **315**. The operating system can provide for interaction with the AT. In one embodiment, a Java runtime is executed on the TCAP, and Java applets communicate with the AT through Java APIs. In another embodiment, a driver is loaded onto the AT, and the on-TCAP Java operating system applets communicate to and through the AT via the driver running on the AT, wherein the driver provides an API through and to which messages may be sent.

After engaging with the AT, the TCAP can provide its memory space to the AT **320**. In one embodiment, the TCAP's memory is mapped and mounted as a virtual disk drive **125** storage **325**. In this manner, the TCAP may be accessed and manipulated as a standard storage device through the AT's operating system. Further, the TCAP and in some cases the AT can determine if the AT is capable of accessing program instructions stored in the TCAP's memory **330**. In one embodiment, the AT's operating system looks to auto-run a specified file from any drive as it mounts. In such an embodiment, the TCAP's primary interface may be specified in such a boot sequence. For example, under windows, an autorun.inf file can specify the opening of a program from the TCAP by the AT; e.g., OPEN=TCAP.EXE.

Many operating systems are capable of at least accessing the TCAP as a USB memory drive **330** and mounting its contents as a drive, which usually becomes accessible in file browsing window **125**. If the TCAP does not mount, the AT's operating system will usually generate an error informing the user of a mounting problem. If the AT is not capable of executing instruction from the TCAP, a determination is made if an appropriate driver is loaded on the AT to access the TCAP **335**. In one embodiment, the TCAP can check to see if an API is running on the AT. For example, the TCAP provide an executable to be launched, e.g., as specified through autorun.inf, and can establish communications through its connection to the AT, e.g., employing TCP/IP communications over the USB port. In such an embodiment, the TCAP can ping the AT for the program, and if an acknowledgement is received, the TCAP has determined that proper drivers and APIs exist. If no such API exists, the TCAP may launch a driver installation program for the AT as through an autorun.inf. In an alternative embodiment, if nothing happens, a user may double click onto an installer program that is stored on the mounted TCAP **342**, **340**. It should be noted, that although the TCAP's memory space may be mounted, certain areas of the TCAP may be inaccessible until there is an authorization. For example, certain areas and content on the TCAP may be encrypted. It should be noted that any such access terminal modules that drive AT and TCAP interaction may be saved onto the TCAP by copying the module to a mounted TCAP. Nevertheless, if the AT is capable of accessing program instructions in TCAP memory **330**, a TCAP driver is loaded on the AT **335**, and/or the user engages a program in the TCAP memory **340**, then the AT can execute program instructions from the TCAP's memory, which allows the TCAP to use the AT's I/O and allowing the user to interface with TCAP facilities **345**. It should be noted that some ATs may not be able to mount the

TCAP at all. In such an instance, the user may have to install the TCAP drivers by downloading them from a server on the Internet, loading them from a diskette or CD, and/or the like. Once the TCAP is engaged to the AT **301**, execution may continue **398**.

TCAP and AT Interaction

FIG. **4** illustrates embodiments accessing the tunneling client access point and server through an access terminal. Upon engaging the TCAP to the AT as described in FIG. **3** **301**, **398**, the user may then go on to access the TCAP and its services **498**. It should be noted that users may access certain unprotected areas of the TCAP once it has been mounted, as described in FIG. **3**. However, to more fully access the TCAP's facilities, the user may be prompted to either login and/or registration window **205***a* to access the TCAP and its services, which may be displayed on the AT **405**. It is important to note that in one embodiment, the execution of the login and/or registration routines are handled by the TCAP's processor. In such an embodiment, the TCAP may run a small Web server providing login facilities, and connect to other Web based services through the AT's connection to the Internet. Further, the TCAP may employ a basic Web browsing core engine by which it may connect to Web services through the AT's connection to a communications network like the Internet. For purposes of security, in one embodiment, the TCAP may connect to a remote server by employing a secure connection, e.g., HTTPS, VPN, and/or the like.

Upon displaying a login window **405**, e.g., **205***a*, the user may select to register to access the TCAP and its services, or they may simply log in by providing security verification. In one example, security authorization may be granted by simply providing a user and password as provided through a registration process. In another embodiment, authorization may be granted through biometric data. For example, the TCAP may integrate a fingerprint and/or heat sensor IC into its housing. Employing such a device, and simply by providing one's finger print by laying your finger to the TCAP's surface, would provide the login facility with authorization if the user's finger print matches one that was stored during the registration process.

If the user does not attempt to login **415**, i.e., if the user wishes to register to use the TCAP and its services, then the TCAP can determine if the AT is online **420**. This may be accomplished in a number of ways. In one embodiment, the TCAP itself may simply ping a given server and if acknowledgement of receipt is received, the TCAP is online. In another embodiment, the TCAP can query for online status by engaging the AT through the installed APIs. If the AT is not online, then the user may be presented with an error message **425**. Thus, if a user does not have a login, and does not have the ability to register, then restricted areas of the TCAP will remain unavailable. Thereafter, flow can continue **498** and the user may have another opportunity to login and/or register. In one embodiment as a login integrity check, the TCAP keeps track of the number of failed attempts to login and/or register and may lock-out all further access if a specified number of failed attempts occurs. In one embodiment, the lockdown may be permanent by erasing all data on the TCAP. In another embodiment, the TCAP will disallow further attempts for a specified period of time.

If the user is attempting to register **415**, and the AT is online **420**, then the user map provide registration information **440** into a screen form **440***a*. Registration information fields may require a user's name, address, email address, credit card information, biometric information (e.g., requiring the user to touch a biometric fingerprint IC on the TCAP), and/or the like. The TCAP may determine if all the information was provided as required for registration and may query backend servers to determine if the user information is unique **445**. If the user did not properly fill out the registration information or if another user is already registered, the TCAP can provided an error message to such effect. Also, both the TCAP and its back-end servers may make log entries tracking such failed attempts for purposes of defending against fraud and/or security breaches. The user may then modify the registration information **440** and again attempt to register. Similarly to the login integrity checks, the TCAP can lockout registration attempts if the user fails to register more than some specified number of times.

Upon providing proper registration information **445** or proper login authentication **415**, the TCAP can query backend servers to see if the user is registered. In one embodiment, such verification may be achieved by sending a query to the servers to check its database for the authorization information and/or for duplicate registrations. The servers would then respond providing an acknowledgment of proper registration and authorization to access data on the backend servers. If the users are not registered on the backend servers **430**, then the TCAP can provide an error message to the user for display on the AT to such effect **435**. In an alternative embodiment, the registration information may be stored on the TCAP itself. In one embodiment, the registration would be maintained in encrypted form. Thus, the user's login information may be checked relative to the information the TCAP itself, and if there is a match, access may be granted, otherwise an error message will be displayed **435**. The TCAP may then continue **498** to operate as if it were just engaged to the AT.

If the user is confirmed to be registered **430**, then the TCAP may provide options for display **453**, **453***a*. Depending on the context and purpose of a particular TCAP, the options may vary. For example, the a screen **453***a* may provide the user with the options to access data either online or offline. The user might simply click on a button and gain secure access to such data that may be decrypted by the TCAP. In one embodiment, the TCAP will determine if the AT is online **455**. If this was already determined **420**, this check **455** may be skipped.

If the AT is online **455**, optionally, the TCAP determines if the user wishes to synchronize the contents of the TCAP with storage facilities at the backend server **470**. In one embodiment, the user may designate that such synchronization is to always take place. If synchronization is specified **470**, then the TCAP will provide and receive updated data to and from the backend servers, overwriting older data with updated versions of the data **475**. If the AT is online **455** and/or after any synchronization **475**, the TCAP may provide the user with all of its service options as authorized by the account and programs available on the TCAP and at the backend server **480**. Once again, these facilities, programs, and/or services may vary greatly depending on the context and deployment requirements of the user. The options to be presented to the user from the TCAP or the TCAP services from the backend server, as displayed through the TCAP onto the AT's display **480**, are myriad and some example embodiments are provided in FIGS. **5-8**. Upon presenting the user with the options, the user is then able to access, execute, store data and programs on the TCAP and on the remote server **485**. All areas of the TCAP and services are then open, including any encrypted data areas.

If the AT is not online **455**, the TCAP may provide options for the user not including online services **460**. In one embodiment, the online options that may be presented on the AT display will be dimmed and/or omitted to reflect the lack of accessibility. However, the user will be able to access, execute, store data and programs on the TCAP, including any encrypted data areas **465**.

TCAP Facilities and Services

FIGS. **5**-**8** illustrate embodiments of facilities, programs, and/or services that the tunneling client access point and server may provide to the user as accessed through an AT. Any particular set of facilities may have a myriad of options. The options and the general nature of the facilities provided on any particular TCAP are dependant upon the requirements of a given set of users. For example, certain groups and/or agencies may require TCAPS to be targeted towards consumer photographs, and may employ TCAPs to further that end. Other groups may require high security facilities, and tailor the TCAPs accordingly. In various environments, an organization may wish to provide a secure infrastructure to all of its agents for securely accessing the organization's data from anywhere and such an organization could tailor the TCAPs contents to reflect and respond to its needs. By providing a generalized infrastructure on the TCAP backend servers and within the TCAP by using a generalized processor, the TCAPs may be deployed in numerous environments.

In one particular embodiment as in FIG. **5**, the TCAP provides facilities to access, process, and store email, files, music, photos and videos through the TCAP. Upon engaging **101** of FIG. **1** the TCAP **130** to an AT **307**, the TCAP will mount and display through the AT's file browser window **125** of FIG. **1**. As has already described, in the case where the AT has no TCAP driver software, the user may double click on the installer software stored on the TCAP **507**. Doing so will launch the installer software from the TCAP's memory to execute on the AT, and the user may be presented with a window to confirm the desire to install the TCAP software onto the AT **507**. Upon confirming the install **507**, the software will install on the AT and the user will be asked to wait as they are apprised of the install progress **509**.

Upon installation, the TCAP front-end software may execute and present the user with various options in various and fanciful interface formats **511**, **460**, **480** of FIG. **4**. In one embodiment, these user interfaces and programs are Java applications that may execute on the AT and a present Java runtime. In an alternative embodiment, a small applet may run on the AT, but all other activities may execute on the TCAP's processor, which would use the AT display only as a display terminal. In the embodiment where the TCAP executes program instructions, the TCAP may be engaged to receive commands and execute by receiving a signal from the access terminal driver instructing it to execute certain program files or, alternatively, looking to default location and executing program instructions. In yet another embodiment, the TCAP may obtain updated interfaces and programs from a backend server for execution either on the TCAP itself and/or the AT; this may be done by synchronization with the backend server and checking for updates of specified files at the backend server. By engaging the user interface, perhaps by clicking on a button to open the TCAP facilities and services **511**, the interface may further unfurl to present options to access said facilities and services **513**. Here, the interface may reflect ownership of the TCAP by providing a welcome screen and showing some resources available to the user; for example, a button entitled "My Stuff" may serve as a mechanism to advance the user to a screen where they may access their personal data store. At this point the user may attempt to login to access their data by engaging an appropriate button, which will take them to a screen that will accept login information **519**. Alternatively, the user may also register if it is their first time using the TCAP by selecting an appropriate button, which will advance the user to a registration screen **515** wherein the user may enter their name, address, credit card information, etc. Upon successfully providing registration information, the user may be prompted for response to further solicitations on a follow-up screen **517**. For example, depending on the services offered for a particular TCAP, the user may be provided certain perks like 5 MB of free online storage on a backend server, free photographic prints, free email access, and/or the like **517**.

After the user is prompted to login **518** and successfully provides proper login information **519**, or after successfully registering **515** and having responded to any solicitations **517**, the user may be provided with general options **521** to access data stored on the TCAP itself **522** or in their online account **520** maintained on a backend server. For example, if the user selects the option to access their online storage **520**, they may be presented with more options to interact with email, files, music, photos and videos that are available online **523**. Perhaps if the user wished to check their email, the user might select to interact with their email, and a screen allowing them to navigate through their email account(s) would be presented **525**. Such online access to data may be facilitated through http protocols whereby the TCAP applications send and receive data through http commands across a communications network interacting with the backend servers and/or other servers. Any received results may be parsed and imbedded in a GUI representation of a Java application. For example, the email facility may run as a Java applet **525** and may employ a POP mail protocol to pull data from a specified mail server to present to the user.

Similarly, many other facilities may be engaged by the user through the TCAP. In one embodiment, the user may drag **508** a file **506** onto a drag-and-drop zone **505** that is presented on the TCAP interface. Upon so doing, various drag-and-drop options may unfurl and present themselves to the user **550**. It should be noted that the file may come from anywhere, i.e., from the AT, the TCAP, and/or otherwise. For example, upon dragging and dropping a graphics file, a user may be prompted with options to order prints, upload the file to an online storage space, save the file to the TCAP's memory space, cancel the action, and/or the like **550**. If the user sends the file for storage, or otherwise wishes to see and manage their data, an interface allowing for such management may be presented **555**. The interface may organize and allow access to general data, picture, and music formats **554**, provide usage statistics (e.g., free space, capacity, used space, etc.) **553**, provide actions to manipulate and organize the data **552**, provide status on storage usage on the TCAP **551** and online **549**, and/or the like.

Should the user engage a user interface element indicating the wish to manipulate their picture data **548**, the TCAP interface will update to allow more specific interaction with the user's photos **557**. In such a screen, the user may select various stored pictures and then indicate a desire to order photo prints by engaging the appropriate user interface element **558**. Should the user indicate their desire for prints **558**, they will be presented with an updated interface allowing the specification of what graphics files they wish to have printed **559**. In one embodiment, the users may drag-and-drop files into a drop zone, or otherwise engage file browsing mechanisms **560** that allow for the selection of desired files. Upon having identified the files for prints **559**, a user may be presented with an interface allowing for the selection of print sizes and quantities **561**. After making such specifications, the user may be required to provide shipping information **563** and information for payments **565**. After providing the billing information to a backend server for processing and approval, the user may be presented with a confirmation interface allowing for editing of the order, providing confirmation of costs, and allowing for submission of a final order for the selected prints **567**. Upon submitting the order, the TCAP will process the files for spooling to a backend server that will accept the order and files, which will be developed as prints and the user's account will be charged accordingly. In one embodiment, all of the above order and image processing operations occur and execute on the TCAP CPU. For example, the TCAP may employ various rendering technologies, e.g., ghostscript, to allow it to read and save PDFs and other media formats.

FIG. **6** goes on to illustrate embodiments and facets of the facilities of FIG. **5**. The TCAP interface allows the user to perform various actions at any given moment. As has already been discussed in FIG. **5**, the user may drag **508** a file **506** onto a drag and drop zone **505** so as to provide the file to the TCAP for further manipulation. As in **550** of FIG. **5**, the user may be presented with various options subsequent to a drag-and-drop operation. Also, the TCAP interface may provide visual feedback that files have been dropped in the drop zone by highlighting the drop zone **505***b*. Should the user wish, they may close the TCAP interface by engaging a close option **633**. Also, the ability to change and/or update their personal information may be accessed through the TCAP interface **616**, which would provide a form allowing the user to update their registration information **630**. In one embodiment, should the user forget their login information, they may request login help **635** and the TCAP will send their authorization information to the last known email address and inform the user of same **640**. Also, the TCAP interface may provide help facilities that may be accessed at any time by simply engaging a help facility user interface element **617**. So doing will provide the user with help screen information as to how to interact with the TCAP's facilities **625**.

Upon providing proper login information **619** and logging-in **619**, the user may be presented with a welcome screen with various options to access their data **621** as has already been discussed in FIG. **5**, **521**. By engaging a user interface element to access online storage **620**, the user may be presented with various options to interact with online storage **623**, **523** of FIG. **5**. Should the user wish to interact with data on the TCAP itself, the user may indicate so by engaging the appropriate user interface option **622**. So doing will provide the user with further options related to data stored on the TCAP **655**. The user may engage an option to view the storage contents **658** and the TCAP interface will provide a listing of the contents **662**, which may be manipulated through selection and drag-and-drop operations with the files.

In one embodiment, the user may order prints of photos **657** from files that are on the TCAP itself. As discussed in FIG. **5**, the user may select files for which they desire prints **660**. Here, the selected files will first be processed by the TCAP in preparation for sending to backend servers and file manipulations **670**. The user may specify various attributes regarding the prints they desire, e.g., the size, number, cropping, red-eye correction, visual effects, and/or the like **661**. In one embodiment, such processing occurs on the TCAP processor, while in other embodiments such processing can take place on the AT or backend server. Once again, the user may provide a shipping address **663**, and make a final review to place the order **667**. Upon committing to the order **667**, the processed files are uploaded to the backend servers that will use the files to generate prints **690**. A confirmation screen may then be provided to the user with an order number and other relevant information **695**.

FIG. **7** goes on to illustrate embodiments and facets of the facilities of FIGS. **5**-**6** as may apply in different environments. As is demonstrated, the look and feel of the TCAP interface is highly malleable and can serve in many environments. FIG. **7** illustrates that even within a single organization, various environments might benefit from TCAPs and services tailored to serve such environments **733***b-d*. In this case TCAPs can serve in consumer **733***b*, industry trade **733***c*, corporate **733***d*, and/or the like environments.

As has already been discussed, initially in any of the environments, after engaging the TCAP to an AT, the user may be prompted to install the TCAP interface **705** and informed of the installation procedure **710**. The user may then be presented with the installed TCAP interface **715**, which may be activated by engaging an interface element to unfurl the interface, e.g., in this case by opening the top to a can of soda **717**. Opening the interface will present the user with various options as **720**, as has already been discussed in FIGS. **5**-**6**. Similarly the user may login **725** or make a selection to register for various TCAP services and provide the requisite information in the provided form **730**. Upon registering and/or logging-in **725**, various options may be presented based upon the configuration of the TCAP. For example, if the TCAP was configured and tailored for consumers, then upon logging in **725** the consumer user might be presented **733***a-b* with various consumer related options **740**. Similarly, if the TCAP were tailored for **733***a, c* the trade industry or **733***a, d* the corporate environment, options specific to the trade industry **770** and corporate environment **760** may be presented.

In one embodiment, an organization wishing to provide TCAPs to consumers might provide options **740** for free music downloads **743**, free Internet radio streaming **748**, free news (e.g., provided through an RSS feed from a server) **766**, free photo printing **750**, free email **740**, free coupons **742**, free online storage **741**, and/or the like. Users could further engage such services (e.g., clicking free music file links for downloading to the TCAP, by ordering prints **750**, etc. For example, the user may select files on the TCAP **750**, select the types of photos they would like to receive **752**, specify a delivery address **754**, confirm the order **756** all of which will result in the TCAP processing the files and uploading them to the backend servers for generation of prints (as has already been discussed in FIGS. **5**-**6**).

In another embodiment, an organization wishing to provide TCAPs to a trade industry might provide options **770** for advertising **780**, events **775**, promotions **772**, and/or the like. It is important to note that information regarding such options may be stored either on the TCAP or at a backend server. In one embodiment, such information may be constantly synchronized from the backend servers to the TCAPs. This would allow an organization to provide updates to the trade industry to all authorized TCAP "key holders." In such an embodiment, the user may be presented with various advertising related materials for the organization, e.g., print, television, outdoor, radio, web, and/or the like **780**. With regard to events, the user may be presented with various related materials for the organization, e.g., trade shows, music regional, sponsorship, Web, and/or the like **775**. With regard to promotions, the user may be presented with various related materials for the organization, e.g., rebates, coupons, premiums, and/or the like **772**.

In another embodiment, an organization wishing to provide TCAPs to those in the corporate environment and might provide options relating to various corporate entities **760**. Selecting any of the corporate entities **760** may provide the user with options to view various reports, presentations, and/or the like, e.g., annual reports, 10K reports, and/or the like **765**. Similarly, the reports may reside on the TCAP and/or the corporate TCAP can act as a security key allowing the user to see the latest corporate related materials from a remote backend server.

FIG. **8** goes on to illustrate embodiments and facets of the facilities of FIGS. **5-7** as may apply in different environments. FIG. **8** illustrates that TCAPs may serve to provide heightened security to any environment. As has been discussed in previous figures, users may engage the TCAP interface **805** to access various options **810**. The TCAP interface is highly adaptable and various services may be presented within it. For example, a stock ticker may be provided as part of the interface in a financial setting **810**. Any number of live data feeds may dynamically update on the face of the interface. Upon logging-in **815** or registering a new account **820**, the user may be informed that communications that are taking place are secured **825**. In one embodiment, various encryption formats may be used by the TCAP to send information securely to the backend servers. It is important to note that in such an embodiment, even if data moving out of the TCAP and across the AT were captured at the AT, such data would not be readable because the data was encrypted by the TCAP's processor. As such, the TCAP acts as a "key" and provides a plug-and-play VPN to users. Such functionality, heretofore, has been very difficult to set up and/or maintain. In this way, all communications, options presented and views of user data are made available only to the TCAP with the proper decryption key. In heightened security environments, display of TCAP data is provided on the screen only in bitmapped format straight to the video memory of the AT and, therefore, is not stored anywhere else on the AT. This decreases the likelihood of capturing sensitive data. As such, the user may access their data on the TCAP and/or online **830** in a secure form whereby the user may navigate and interact with his/her data and various services **835** in a secure manner.

Tunneling Client Access Point Server Controller

FIG. **9** illustrates one embodiment incorporated into a tunneling client access point server (TCAPS) controller **901**. In this embodiment, the TCAP controller **901** may serve to process, store, search, serve, identify, instruct, generate, match, and/or update data in conjunction with a TCAP (see FIG. **10** for more details on the TCAP). TCAPS act as backend servers to TCAPs, wherein TCAPS provide storage and/or processing resources to great and/or complex for the TCAP to service itself. In effect, the TCAPS transparently extend the capacity of a TCAP.

In one embodiment, the TCAPS controller **901** may be connected to and/or communicate with entities such as, but not limited to: one or more users from user input devices **911**; peripheral devices **912**; and/or a communications network **913**. The TCAPS controller may even be connected to and/or communicate with a cryptographic processor device **928**.

A TCAPS controller **901** may be based on common computer systems that may comprise, but are not limited to, components such as: a computer systemization **902** connected to memory **929**.

Computer Systemization

A computer systemization **902** may comprise a clock **930**, central processing unit (CPU) **903**, a read only memory (ROM) **906**, a random access memory (RAM) **905**, and/or an interface bus **907**, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus **904**. Optionally, a cryptographic processor **926** may be connected to the system bus. The system clock typically has a crystal oscillator and provides a base signal. The clock is typically coupled to the system bus and various clock multipliers that will increase or decrease the base operating frequency for other components interconnected in the computer systemization. The clock and various components in a computer systemization drive signals embodying information throughout the system. Such transmission and reception of signals embodying information throughout a computer systemization may be commonly referred to as communications. These communicative signals may further be transmitted, received, and the cause of return and/or reply signal communications beyond the instant computer systemization to: communications networks, input devices, other computer systemizations, peripheral devices, and/or the like. Of course, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

The CPU comprises at least one high-speed data processor adequate to execute program modules for executing user and/or system-generated requests. The CPU may be a microprocessor such as AMD's Athlon, Duron and/or Opteron; IBM and/or Motorola's PowerPC; Intel's Celeron, Itanium, Pentium and/or Xeon; and/or the like processor(s). The CPU interacts with memory through signal passing through conductive conduits to execute stored program code according to conventional data processing techniques. Such signal passing facilitates communication within the TCAPS controller and beyond through various interfaces. Should processing requirements dictate a greater amount speed, mainframe and super computer architectures may similarly be employed.

Interface Adapters

Interface bus(ses) **907** may accept, connect, and/or communicate to a number of interface adapters, conventionally although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) **908**, storage interfaces **909**, network interfaces **910**, and/or the like. Optionally, cryptographic processor interfaces **927** similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. Interface adapters conventionally connect to the interface bus via a slot architecture. Conventional slot architectures may be employed, such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended) Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI(X)), Personal Computer Memory Card International Association (PCMCIA), and/or the like.

Storage interfaces **909** may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices **914**, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE), Institute of Electrical and Electronics

Engineers (IEEE) 1394, fiber channel, Small Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

Network interfaces **910** may accept, communicate, and/or connect to a communications network **913**. Network interfaces may employ connection protocols such as, but not limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T, and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the like. A communications network may be any one and/or the combination of the following: a direct interconnection; the Internet; a Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network (WAN); a wireless network (e.g., employing protocols such as, but not limited to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A network interface may be regarded as a specialized form of an input output interface. Further, multiple network interfaces **910** may be used to engage with various communications network types **913**. For example, multiple network interfaces may be employed to allow for the communication over broadcast, multicast, and/or unicast networks. Input Output interfaces (I/O) **908** may accept, communicate, and/or connect to user input devices **911**, peripheral devices **912**, cryptographic processor devices **928**, and/or the like. I/O may employ connection protocols such as, but not limited to: Apple Desktop Bus (ADB); Apple Desktop Connector (ADC); audio: analog, digital, monaural, RCA, stereo, and/or the like; IEEE 1394a-b; infrared; joystick; keyboard; midi; optical; PC AT; PS/2; parallel; radio; serial; USB; video interface: BNC, composite, digital, Digital Visual Interface (DVI), RCA, S-Video, VGA, and/or the like; wireless; and/or the like. A common output device is a video display, which typically comprises a Cathode Ray Tube (CRT) or Liquid Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry and cable) that accepts signals from a video interface. The video interface composites information generated by a computer systemization and generates video signals based on the composited information in a video memory frame. Typically, the video interface provides the composited video information through a video connection interface that accepts a video display interface (e.g., a DVI connector accepting a DVI display cable).

User input devices **911** may be card readers, dongles, finger print readers, gloves, graphics tablets, joysticks, keyboards, mouse (mice), trackballs, trackpads, retina readers, and/or the like.

Peripheral devices **912** may be connected and/or communicate to I/O and/or other facilities of the like such as network interfaces, storage interfaces, and/or the like. Peripheral devices may be audio devices, cameras, dongles (e.g., for copy protection, ensuring secure transactions with a digital signature, and/or the like), external processors (for added functionality), goggles, microphones, monitors, network interfaces, printers, scanners, storage devices, video devices, visors, and/or the like.

It should be noted that although user input devices and peripheral devices may be employed, the TCAPS controller may be embodied as an embedded, dedicated, and/or headless device, wherein access would be provided over a network interface connection.

Cryptographic units such as, but not limited to, microcontrollers, processors **926**, interfaces **927**, and/or devices **928** may be attached, and/or communicate with the TCAPS controller. A MC68HC16 microcontroller, commonly manufactured by Motorola Inc., may be used for and/or within cryptographic units. Equivalent microcontrollers and/or processors may also be used. The MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the 16 MHz configuration and requires less than one second to perform a 512-bit RSA private key operation. Cryptographic units support the authentication of communications from interacting agents, as well as allowing for anonymous transactions. Cryptographic units may also be configured as part of CPU. Other commercially available specialized cryptographic processors include VLSI Technology's 33 MHz 6868 or Semaphore Communications' 40 MHz Roadrunner 184.

Memory

Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory **929**. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be understood that a TCAPS controller and/or a computer systemization may employ various forms of memory **929**. For example, a computer systemization may be configured wherein the functionality of on-chip CPU memory (e.g., registers), RAM, ROM, and any other storage devices are provided by a paper punch tape or paper punch card mechanism; of course such an embodiment would result in an extremely slow rate of operation. In a typical configuration, memory **929** will include ROM **906**, RAM **905**, and a storage device **914**. A storage device **914** may be any conventional computer system storage. Storage devices may include a drum; a (fixed and/or removable) magnetic disk drive; a magneto-optical drive; an optical drive (i.e., CD ROM/RAM/Recordable (R), ReWritable (RW), DVD R/RW, etc.); and/or other devices of the like. Thus, a computer systemization generally requires and makes use of memory.

Module Collection

The memory **929** may contain a collection of program and/or database modules and/or data such as, but not limited to: operating system module(s) **915** (operating system); information server module(s) **916** (information server); user interface module(s) **917** (user interface); Web browser module(s) **918** (Web browser); database(s) **919**; cryptographic server module(s) **920** (cryptographic server); TCAPS module(s) **935**; and/or the like (i.e., collectively a module collection). These modules may be stored and accessed from the storage devices and/or from storage devices accessible through an interface bus. Although non-conventional software modules such as those in the module collection, typically, are stored in a local storage device **914**, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through a communications network, ROM, various forms of memory, and/or the like.

Operating System

The operating system module **915** is executable program code facilitating the operation of a TCAPS controller. Typically, the operating system facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as Apple Macintosh OS X (Server), AT&T Plan 9, Be OS, Linux, Unix, and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Apple Macintosh OS, Microsoft DOS, Palm OS, Windows 2000/2003/3.1/95/98/CE/Millenium/NT/XP (Server), and/or the like. An operating system may communicate to and/or with other modules in a module collection, including itself, and/or the like. Most frequently, the operating system communicates with other program modules, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the interaction with communications networks, data, I/O, peripheral devices, program modules, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the TCAPS controller to communicate with other entities through a communications network **913**. Various communication protocols may be used by the TCAPS controller as a subcarrier transport mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the like.

Information Server

An information server module **916** is stored program code that is executed by the CPU. The information server may be a conventional Internet information server such as, but not limited to Apache Software Foundation's Apache, Microsoft's Internet Information Server, and/or the. The information server may allow for the execution of program modules through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-) C (++), Common Gateway Interface (CGI) scripts, Java, JavaScript, Practical Extraction Report Language (PERL), Python, WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program modules. After a Domain Name System (DNS) resolution portion of an HTTP request is resolved to a particular information server, the information server resolves requests for information at specified locations on a TCAPS controller based on the remainder of the HTTP request. For example, a request such as http://123.124.125.126/myInformation.html might have the IP portion of the request "123.124.125.126" resolved by a DNS server to an information server at that IP address; that information server might in turn further parse the http request for the "/myInformation.html" portion of the request and resolve it to a location in memory containing the information "myInformation.html." Additionally, other information serving protocols may be employed across various ports, e.g., FTP communications across port **21**, and/or the like. An information server may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the TCAPS database **919**, operating systems, other program modules, user interfaces, Web browsers, and/or the like.

Access to TCAPS database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the TCAP. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the TCAPS as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

Also, an information server may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

User Interface

A user interface module **917** is stored program code that is executed by the CPU. The user interface may be a conventional graphic user interface as provided by, with, and/or atop operating systems and/or operating environments such as Apple Macintosh OS, e.g., Aqua, Microsoft Windows (NT/XP), Unix X Windows (KDE, Gnome, and/or the like), and/or the like. The user interface may allow for the display, execution, interaction, manipulation, and/or operation of program modules and/or system facilities through textual and/or graphical facilities. The user interface provides a facility through which users may affect, interact, and/or operate a computer system. A user interface may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the user interface communicates with operating systems, other program modules, and/or the like. The user interface may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Web Browser

A Web browser module **918** is stored program code that is executed by the CPU. The Web browser may be a conventional hypertext viewing application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web browsing may be supplied with 128 bit (or greater) encryption by way of HTTPS, SSL, and/or the like. Some Web browsers allow for the execution of program modules through facilities such as Java, JavaScript, ActiveX, and/or the like. Web browsers and like information access tools may be integrated into PDAs, cellular telephones, and/or other mobile devices. A Web browser may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the Web browser communicates with information servers, operating systems, integrated program modules (e.g., plug-ins), and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. Of course, in place of a Web browser and information server, a combined application may be developed to perform similar functions of both. The combined application would similarly affect the obtaining and the provision of information to users, user agents, and/or the like from TCAPS enabled nodes. The combined application may be nugatory on systems employing standard Web browsers.

TCAPS Database

A TCAPS database module **919** may be embodied in a database and its stored data. The database is stored program code, which is executed by the CPU; the stored program code portion configuring the CPU to process the stored data. The database may be a conventional, fault tolerant, relational, scalable, secure database such as Oracle or Sybase. Relational databases are an extension of a flat file. Relational databases consist of a series of related tables. The tables are interconnected via a key field. Use of the key field allows the combination of the tables by indexing against the key field; i.e., the key fields act as dimensional pivot points for combining information from various tables. Relationships generally identify links maintained between tables by matching primary keys. Primary keys represent fields that uniquely identify the rows of a table in a relational database. More precisely, they uniquely identify rows of a table on the "one" side of a one-to-many relationship.

Alternatively, the TCAPS database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In another alternative, an object-oriented database may be used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can include a number of object collections that are grouped and/or linked together by common attributes; they may be related to other object collections by some common attributes. Object-oriented databases perform similarly to relational databases with the exception that objects are not just pieces of data but may have other types of functionality encapsulated within a given object. If the TCAPS database is implemented as a data-structure, the use of the TCAPS database may be integrated into another module such as the TCAPS module. Also, the database may be implemented as a mix of data structures, objects, and relational structures. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated. In one embodiment, the database module **919** includes three tables **919***a-c*. A user accounts table **919***a* includes fields such as, but not limited to: a user name, user address, user authorization information (e.g., user name, password, biometric data, etc.), user credit card, organization, organization account, TCAP unique identifier, account creation data, account expiration date; and/or the like. In one embodiment, user accounts may be activated only for set amounts of time and will then expire once a specified date has been reached. An user data table **919***b* includes fields such as, but not limited to: a TCAP unique identifier, backup image, data store, organization account, and/or the like. A user programs table **919***c* includes fields such as, but not limited to: system programs, organization programs, programs to be synchronized, and/or the like. In one embodiment, user programs may contain various user interface primitives, which may serve to update TCAPs. Also, various accounts may require custom database tables depending upon the environments and the types of TCAPs a TCAPS may need to serve. It should be noted that any unique fields may be designated as a key field throughout. In an alternative embodiment, these tables have been decentralized into their own databases and their respective database controllers (i.e., individual database controllers for each of the above tables). Employing standard data processing techniques, one may further distribute the databases over several computer systemizations and/or storage devices. Similarly, configurations of the decentralized database controllers may be varied by consolidating and/or distributing the various database modules **919***a-c*. The TCAPS may be configured to keep track of various settings, inputs, and parameters via database controllers.

A TCAPS database may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAPS database communicates with a TCAPS module, other program modules, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.

Cryptographic Server

A cryptographic server module **920** is stored program code that is executed by the CPU **903**, cryptographic processor **926**, cryptographic processor interface **927**, cryptographic processor device **928**, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic module; however, the cryptographic module, alternatively, may run on a conventional CPU. The cryptographic module allows for the encryption and/or decryption of provided data. The cryptographic module allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic module may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic module will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like. Employing such encryption security protocols, the TCAPS may encrypt all incoming and/or outgoing communications and may serve as node within a virtual private network (VPN) with a wider communications network. The cryptographic module facilitates the process of "security authorization" whereby access to a resource is inhibited by a security protocol wherein the cryptographic module effects authorized access to the secured resource. In addition, the cryptographic module may provide unique identifiers of content, e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A cryptographic module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. The cryptographic module supports encryption schemes allowing for the secure transmission of information across a communications network to enable a TCAPS module to engage in secure transactions if so desired. The cryptographic module facilitates the secure accessing of resources on TCAPS and facilitates the access of secured resources on remote systems; i.e., it may act as a client and/or server of secured resources. Most frequently, the cryptographic module communicates with information servers, operating systems, other program modules, and/or the like. The cryptographic module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

TCAPS

A TCAPS module **935** is stored program code that is executed by the CPU. The TCAPS affects accessing, obtaining and the provision of information, services, transactions, and/or the like across various communications networks. The TCAPS enables TCAP users to simply access data and/or services across a communications network in a secure manner. The TCAPS extends the storage and processing capacities and capabilities of TCAPs. The TCAPS coordinates with the TCAPS database to identify interassociated items in the generation of entries regarding any related information. A TCAPS module enabling access of information between nodes may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective-) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the TCAPS server employs a cryptographic server to encrypt and decrypt communications. A TCAPS module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAPS module communicates with a TCAPS database, operating systems, other program modules, and/or the like. The TCAPS may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Distributed TCAP

The structure and/or operation of any of the TCAPS node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the module collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

The module collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program modules in the program module collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program module instances and controllers working in concert may do so through standard data processing communication techniques.

The configuration of the TCAPS controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program modules, results in a more distributed series of program modules, and/or results in some combination between a consolidated and distributed configuration, data may be communicated, obtained, and/or provided. Instances of modules consolidated into a common code base from the program module collection may communicate, obtain, and/or provide data. This may be accomplished through intra-application data processing communication techniques such as, but not limited to: data referencing (e.g., pointers), internal messaging, object instance variable communication, shared memory space, variable passing, and/or the like.

If module collection components are discrete, separate, and/or external to one another, then communicating, obtaining, and/or providing data with and/or to other module components may be accomplished through inter-application data processing communication techniques such as, but not limited to: Application Program Interfaces (API) information passage; (distributed) Component Object Model ((D) COM), (Distributed) Object Linking and Embedding ((D) OLE), and/or the like), Common Object Request Broker Architecture (CORBA), process pipes, shared files, and/or the like. Messages sent between discrete module components for inter-application communication or within memory spaces of a singular module for intra-application communication may be facilitated through the creation and parsing of a grammar. A grammar may be developed by using standard development tools such as lex, yacc, and/or the like, which allow for grammar generation and parsing functionality, which in turn may form the basis of communication messages within and between modules. Again, the configuration will depend upon the context of system deployment.

Tunneling Client Access Point Controller

FIG. **10** illustrates one embodiment incorporated into a tunneling client access point (TCAP) controller **1001**. Much of the description of the TCAPS of FIG. **9** applies to the TCAP, and as such, the disclosure focuses more upon the variances exhibited in the TCAP. In this embodiment, the TCAP controller **1001** may serve to process, store, search, identify, instruct, generate, match, and/or update data within itself, at a TCAPS, and/or through an AT.

The first and foremost difference between the TCAP and the TCAPS is that the TCAP is very small as was shown **130** of FIG. **1**. The TCAP may be packaged in plugin sticks, often, smaller than the size of a human thumb. In one embodiment, a TCAP may be hardened for military use. In such an embodiment, the shell **1001** may be composed of metal, and/or other durable composites. Also, components within may be shielded from radiation.

In one embodiment, the TCAP controller **1001** may be connected to and/or communicate with entities such as, but not limited to: one or more users from an access terminal **1011***b*. The access terminal itself may be connected to peripherals such as user input devices (e.g., keyboard **1012***a*, mouse **1012***b*, etc.); and/or a communications network **1013** in manner similar to that described in FIG. **9**.

A TCAP controller **1001** may be based on common computer systems components that may comprise, but are not limited to, components such as: a computer systemization **1002** connected to memory **1029**. Optionally, the TCAP controller **1001** may convey information **1058**, produce output through an output device **1048**, and obtain input from control device **1018**.

Control Device

The control device **1018** may be optionally provided to accept user input to control access to the TCAP controller. In one embodiment, the control device may provide a keypad **1028**. Such a keypad would allow the user to enter passwords, personal identification numbers (PIN), and/or the like.

In an alternative embodiment, the control device may include a security device **1038**. In one embodiment, the security device is a fingerprint integrated circuit (fingerprint IC) that provides biometric fingerprint information such as, but not limited to AuthenTec Inc.'s FingerLoc™ AF-S2™. Either a fingerprint IC and/or other biometric device will provide biometric validation information that may be used to confirm the identity of a TCAP user and ensure that transactions are legitimate. In alternative embodiments, a simple button, heat sensor, and/or other type of user input functionality may be provided solely and/or in concert with other types of control device types. The control device may be connected to the I/O interface, the system bus, or the CPU directly.

The output device **1048** is used to provide status information to the user. In one alternative embodiment, the output device is an LCD panel capable of providing alpha numeric and/or graphic displays. In an alternative embodiment, the output device may be a speaker providing audible signals indicating errors and/or actually streaming information that is audible to the user, such as voice alerts. The output device may be connected to the I/O interface, the system bus, or the CPU directly.

The conveyance information **1058** component of the TCAP controller may include any number of indicia representing the TCAP's source on the cover **1001**. Source conveying indicia may include, but is not limited to: an owner name **1059** for readily verifying a TCAP user; a photo of the owner **1060** for readily verifying a TCAP controller owner; mark designating the source that issued the TCAP **1061**, **1001** such as a corporate logo, and/or the like; fanciful design information **1062** for enhancing the visual appearance of the TCAP; and/or the like. It should be noted that the conveyance information **11421** may be positioned anywhere on the cover **1189**.

Computer Systemization

A computer systemization **1002** may comprise a clock **1030**, central processing unit (CPU) **1003**, a read only memory (ROM) **1006**, a random access memory (RAM) **1005**, and/or an interface bus **1007**, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus **1004**. Optionally the computer systemization may be connected to an internal power source **1086**. Optionally, a cryptographic processor **1026** may be connected to the system bus. The system clock typically has a crystal oscillator and provides a base signal. Of course, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

The CPU comprises at least one low-power data processor adequate to execute program modules for executing user and/or system-generated requests. The CPU may be a microprocessor such as ARM's Application Cores, Embedded Cores, Secure Cores; Motorola's DragonBall; and/or the like processor(s).

Power Source

The power source **1086** may be of any standard form for powering small electronic circuit board devices such as but not limited to: alkaline, lithium hydride, lithium ion, nickel cadmium, solar cells, and/or the like. In the case of solar cells, the case provides an aperture through which the solar cell protrudes are to receive photonic energy. The power cell **1086** is connected to at least one of the interconnected subsequent components of the TCAP thereby providing an electric current to all subsequent components. In one example, the power cell **1086** is connected to the system bus component **1004**. In an alternative embodiment, an outside power source **1086** is provided through a connection across the I/O **1008** interface. For example, a USB and/or IEEE 1394 connection carries both data and power across the connection and is therefore a suitable source of power.

Interface Adapters

Interface bus(ses) **1007** may accept, connect, and/or communicate to a number of interface adapters, conventionally although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) **1008**, storage interfaces **1009**, network interfaces **1010**, and/or the like. Optionally, cryptographic processor interfaces **1027** similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. In one embodiment, the interface bus provides I/O **1008** via a USB port. In an alternative embodiment, the interface bus provides I/O via an IEEE 1394 port. In an alternative embodiment, wireless transmitters are employed by interfacing wireless protocol integrated circuits (ICs) for I/O via the interface bus **1007**.

Storage interfaces **1009** may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices **1014**, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to a flash memory connector, and/or the like. In one embodiment, an optional network interface may be provide **1010**.

Input Output interfaces (I/O) **1008** may accept, communicate, and/or connect to an access terminal **1011***b*. I/O may employ connection protocols such as, but not limited to: Apple Desktop Bus (ADB); Apple Desktop Connector (ADC); IEEE 1394a-b; infrared; PC AT; PS/2; parallel; radio; serial; USB, and/or the like; wireless component; and/or the like.

Wireless Component

In one embodiment a wireless component may comprise a Bluetooth chip disposed in communication with a transceiver **1043** and a memory **1029** through the interface bus **1007** and/or system bus **1004**. The transceiver may be either external to the Bluetooth chip, or integrated within the Bluetooth chip itself. The transceiver is a radio frequency (RF) transceiver operating in the range as required for Bluetooth transmissions. Further, the Bluetooth chip **1044** may integrate an input/output interface (I/O) **1066**. The Bluetooth chip and its I/O may be configured to interface with the TCAP controller through the interface bus, the system buss, and/or directly with the CPU. The I/O may be used to interface with other components such as an access terminal **1011***b* equipped with similar wireless capabilities. In one embodiment, the TCAP may optionally interconnect wirelessly with a peripheral device **912** and/or a control device **911** of FIG. **9**. In one example embodiment, the I/O may be based on serial line technologies, a universal serial bus (USB) protocol, and/or the like. In an alternative embodiment, the I/O may be based on the ISO 7816-3 standard. It should be noted that the Bluetooth chip in an alternative embodiment may be replaced with an IEEE 802.11b wireless chip. In another embodiment, both a Bluetooth chip and an IEEE 802.11b wireless chip may be used to communicate and or bridge communications with respectively enabled devices. It should further be noted that the transceiver **1043** may be used to wirelessly communicate with other devices powered by Bluetooth chips and/or IEEE 802.11b chips and/or the like. The ROM can provide a basic instruction set enabling the Bluetooth chip to use its I/O to communicate with other components. A number of Bluetooth chips are commercially available, and may be used as a Bluetooth chip in the wireless component, such as, but not limited to, CSR's BlueCore line of chips. If IEEE 802.11b functionality is required, a number of chips are commercially available for the wireless component as well.

Cryptographic units such as, but not limited to, microcontrollers, processors **1026**, and/or interfaces **1027** may be attached, and/or communicate with the TCAP controller. A Secure Core component commonly manufactured by ARM, Inc. and may be used for and/or within cryptographic units.

Memory

Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory **1029**. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be understood that a TCAP controller and/or a computer systemization may employ various forms of memory **1029**. In a typical configuration, memory **1029** will include ROM **1006**, RAM **1005**, and a storage device **1014**. A storage device **1014** may be any conventional computer system storage. Storage devices may include flash memory, micro hard drives, and/or the like.

Module Collection

The memory **1029** may contain a collection of program and/or database modules and/or data such as, but not limited to: operating system module(s) **1015** (operating system); information server module(s) **1016** (information server); user interface module(s) **1017** (user interface); Web browser module(s) **1018** (Web browser); database(s) **1019**; cryptographic server module(s) **1020** (cryptographic server); access terminal module **1021**; TCAP module(s) **1035**; and/or the like (i.e., collectively a module collection). These modules may be stored and accessed from the storage devices and/or from storage devices accessible through an interface bus. Although non-conventional software modules such as those in the module collection, typically, are stored in a local storage device **1014**, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through an access terminal, communications network, ROM, various forms of memory, and/or the like. In one embodiment, all data stored in memory is encrypted by employing the cryptographic server **1020** as described in further detail below. In one embodiment, the ROM contains a unique TCAP identifier. For example, the TCAP may contain a unique digital certificate, number, and/or the like, which may be used for purposes of verification and encryption across a network and/or in conjunction with a TCAPS.

Operating System

The operating system module **1015** is executable program code facilitating the operation of a TCAP controller. Typically, the operating system facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as Linux, and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Java runtime OS, and/or the like. An operating system may communicate to and/or with other modules in a module collection, including itself, and/or the like. Most frequently, the operating system communicates with other program modules, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the interaction with an access terminal, communications networks, data, I/O, peripheral devices, program modules, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the TCAP controller to communicate with other entities through an access terminal. Various communication protocols may be used by the TCAP controller as a subcarrier transport mechanism for interaction, such as, but not limited to: TCP/IP, USB, and/or the like.

Information Server

An information server module **1016** is stored program code that is executed by the CPU. The information server may be a conventional Internet information server such as, but not limited to Apache Software Foundation's Apache, and/or the like. The information server may allow for the execution of program modules through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-) C (++), Common Gateway Interface (CGI) scripts, Java, JavaScript, Practical Extraction Report Language (PERL), Python, WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program modules. An information server may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the TCAP database **1019**, operating systems, other program modules, user interfaces, Web browsers, and/or the like.

Access to TCAP database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the TCAP. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the TCAP as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

Also, an information server may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

User Interface

A user interface module **1017** is stored program code that is executed by the CPU. The user interface may be a conventional graphic user interface as provided by, with, and/or atop operating systems and/or operating environments such as Apple Macintosh OS, e.g., Aqua, Microsoft Windows (NT/XP), Unix X Windows (KDE, Gnome, and/or the like), and/or the like. The TCAP may employ code natively compiled for various operating systems, or code compiled using Java. The user interface may allow for the display, execution, interaction, manipulation, and/or operation of program modules and/or system facilities through textual and/or graphical facilities. The user interface provides a facility through which users may affect, interact, and/or operate a computer system. A user interface may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the user interface communicates with operating systems, other program modules, and/or the like. The user interface may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Web Browser

A Web browser module **1018** is stored program code that is executed by the CPU. A small-scale embedded Web browser may allow the TCAP to access and communicate with an attached access terminal, and beyond across a communications network. An example browser is Blazer, Opera, FireFox, etc. A browsing module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. Of course, in place of a Web browser and information server, a combined application may be developed to perform similar functions of both. The combined application would similarly affect the obtaining and the provision of information to users, user agents, and/or the like from TCAP enabled nodes. The combined application may be nugatory on systems employing standard Web browsers.

TCAP Database

A TCAP database module **1019** may be embodied in a database and its stored data. The database is stored program code, which is executed by the CPU; the stored program code portion configuring the CPU to process the stored data. In one embodiment, the TCAP database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. If the TCAP database is implemented as a data-structure, the use of the TCAP database may be integrated into another module such as the TCAP module. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated. In one embodiment, the database module **1019** includes three tables **1019***a-c*. A user accounts table **1019***a* includes fields such as, but not limited to: a user name, user address, user authorization information (e.g., user name, password, biometric data, etc.), user credit card, organization, organization account, TCAP unique identifier, account creation data, account expiration date; and/or the like. In one embodiment, user accounts may be activated only for set amounts of time and will then expire once a specified date has been reached. An user data table **1019***b* includes fields such as, but not limited to: a TCAP unique identifier, backup image, data store, organization account, and/or the like. In one embodiment, the entire TCAP memory **1029** is processes into an image and spooled to a TCAPS for backup storage. A user programs table **1019***c* includes fields such as, but not limited to: system programs, organization programs, programs to be synchronized, and/or the like. It should be noted that any unique fields may be designated as a key field throughout. In an alternative embodiment, these tables have been decentralized into their own databases and their respective database controllers (i.e., individual database controllers for each of the above tables). Employing standard data processing techniques, one may further distribute the databases over several computer systemizations and/or storage devices. Similarly, configurations of the decentralized database controllers may be varied by consolidating and/or distributing the various database modules **1019***a-c*. The TCAP may be configured to keep track of various settings, inputs, and parameters via database controllers.

A TCAP database may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAP database communicates with a TCAP module, other program modules, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.

Cryptographic Server

A cryptographic server module **1020** is stored program code that is executed by the CPU **1003**, cryptographic processor **1026**, cryptographic processor interface **1027**, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic module; however, the cryptographic module, alternatively, may run on a conventional CPU. The cryptographic module allows for the encryption and/or decryption of provided data. The cryptographic module allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic module may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic module will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like. The cryptographic module facilitates the process of "security authorization" whereby access to a resource is inhibited by a security protocol wherein the cryptographic module effects authorized access to the secured resource. In addition, the cryptographic module may provide unique identifiers of content, e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A cryptographic module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. The cryptographic module supports encryption schemes allowing for the secure transmission of information across a communications network to enable a TCAP module to engage in secure transactions if so desired. The cryptographic module facilitates the secure accessing of resources on TCAP and facilitates the access of secured resources on remote systems; i.e., it may act as a client and/or server of secured resources. Most frequently, the cryptographic module communicates with information servers, operating systems, other program modules, and/or the like. The cryptographic module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. In one embodiment, the TCAP employs the cryptographic server to encrypt all data stored in memory **1029** based on the TCAP's unique ID and user's authorization information. In another embodiment, the TCAP employs the cryptographic server to encrypt all data sent through the access terminal based in the TCAP's unique ID and user's authorization information.

TCAP

A TCAP module **1035** is stored program code that is executed by the CPU. The TCAP affects accessing, obtaining and the provision of information, services, storage, transactions, and/or the like within its memory and/or across various communications networks. The TCAP enables users to simply access data and/or services from any location where an access terminal is available. It provides secure, extremely low powerful and ultra portable access to data and services that were heretofore impossible. The TCAP coordinates with the TCAP database to identify interassociated items in the generation of entries regarding any related information. A TCAP module enabling access of information between nodes may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective-) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the TCAP server employs a cryptographic server to encrypt and decrypt communications. A TCAP module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAP module communicates with a TCAP database, a TCAP access terminal module **1021** running on an access terminal **1011***b*, operating systems, other program modules, and/or the like. The TCAP may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Access Terminal Module

An access terminal module **1021** is stored program code that is executed by a CPU. In one embodiment, the TCAP allows the access terminal **1011***b* to access its memory **1029** across its I/O **1008** and the access terminal executes the module. The access terminal module affects accessing, obtaining and the provision of information, services, storage, transactions, and/or the like within the TCAP's and access terminal's memory and/or across various communications networks. The access terminal module **1021** acts as a bridge through which the TCAP can communicate with communications network, and through which users may interact with the TCAP by using the I/O of the access terminal. The access terminal module coordinates with the TCAP module **1035** to send data and communications back and forth. A access terminal module enabling access of information between the TCAP and access terminal may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective-) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the access terminal module is compiled for target access terminal platform, e.g., for Windows. In an alternative embodiment, a processor independent approach is taken, e.g., Java is used, so that the access terminal module will run on multiple platforms. In another embodiment, the TCAP server employs a cryptographic server to encrypt and decrypt communications as between it, the TCAP, and outside servers. A access terminal module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the access terminal module communicates with a TCAP, other program modules, and/or the like. The access terminal module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Distributed TCAP

The structure and/or operation of any of the TCAP node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the module collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

The module collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program modules in the program module collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program module instances and controllers working in concert may do so through standard data processing communication techniques.

The configuration of the TCAP controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program modules, results in a more distributed series of program modules, and/or results in some combination between a consolidated and distributed configuration, data may be communicated, obtained, and/or provided. Instances of modules consolidated into a common code base from the program module collection may communicate, obtain, and/or provide data. This may be accomplished through intra-application data processing communication techniques such as, but not limited to: data referencing (e.g., pointers), internal messaging, object instance variable communication, shared memory space, variable passing, and/or the like.

If module collection components are discrete, separate, and/or external to one another, then communicating, obtaining, and/or providing data with and/or to other module components may be accomplished through inter-application data processing communication techniques such as, but not limited to: Application Program Interfaces (API) information passage; (distributed) Component Object Model ((D) COM), (Distributed) Object Linking and Embedding ((D) OLE), and/or the like), Common Object Request Broker Architecture (CORBA), process pipes, shared files, and/or the like. Messages sent between discrete module components for inter-application communication or within memory spaces of a singular module for intra-application communication may be facilitated through the creation and parsing of a grammar. A grammar may be developed by using standard development tools such as lex, yacc, and/or the like, which allow for grammar generation and parsing functionality, which in turn may form the basis of communication messages within and between modules. Again, the configuration will depend upon the context of system deployment.

The entirety of this disclosure (including the Cover Page, Title, Headings, Field, Background, Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract, Figures, and otherwise) shows by way of illustration various embodiments in which the claimed inventions may be practiced. The advantages and features of the disclosure are of a representative sample of embodiments only, and are not exhaustive and/or exclusive. They are presented only to assist in understanding and teach the claimed principles. It should be understood that they are not representative of all claimed inventions. As such, certain aspects of the disclosure have not been discussed herein. That alternate embodiments may not have been presented for a specific portion of the invention or that further undescribed alternate embodiments may be available for a portion is not to be considered a disclaimer of those alternate embodiments. It will be appreciated that many of those undescribed embodiments incorporate the same principles of the invention and others are equivalent. Thus, it is to be understood that other embodiments may be utilized and functional, logical, organizational, structural and/or topological modifications may be made without departing from the scope and/or spirit of the disclosure. As such, all examples and/or embodiments are deemed to be non-limiting throughout this disclosure. Also, no inference should be drawn regarding those embodiments discussed herein relative to those not discussed herein other than for purposes of space and reducing repetition. For instance, it is to be understood that the logical and/or topological structure of any combination of any program modules (a module collection), other components and/or any present feature sets as described in the figures and/or throughout are not limited to a fixed operating order and/or arrangement, but rather, any disclosed order is exemplary and all equivalents, regardless of order, are contemplated by the disclosure. Furthermore, it is to be understood that such features are not limited to serial execution, but rather, any number of threads, processes, services, servers, and/or the like that may execute asynchronously, simultaneously, synchronously, and/or the like are contemplated by the disclosure. As such, some of these features may be mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some features are applicable to one aspect of the invention, and inapplicable to others. In addition, the disclosure includes other inventions not presently claimed. Applicant reserves all rights in those presently unclaimed inventions including the right to claim such inventions, file additional applications, continuations, continuations in part, divisions, and/or the like thereof. As such, it should be understood that advantages, embodiments, examples, functional, features, logical, organizational, structural, topological, and/or other aspects of the disclosure are not to be considered limitations on the disclosure as defined by the claims or limitations on equivalents to the claims.

What is claimed is:

1. A portable device configured to communicate with a terminal comprising a processor and an output component and with a communications network comprising a plurality of communications network nodes, the portable device comprising:

(a) a communication interface configured to enable the transmission of communications to the terminal;

(b) a network interface configured to enable the transmission of communications between the portable device and a communications network node;

(c) a processor; and

(d) a memory having executable program code stored thereon, including:

(1) first program code which, when executed, is configured to present an interactive user interface on the terminal output component,

(2) second program code which, when executed by the portable device processor, is configured to facilitate communications between the portable device and the communications network node; and

(3) third program code which, when executed by the portable device processor in response to user interaction with the interactive user interface, is configured to cause a communication to be transmitted through the portable device network interface to the communications network node.

2. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate verification of the portable device.

3. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate the download of data to the portable device.

4. The portable device according to claim **3**, wherein the portable device is configured to transmit to the terminal data downloaded from the communications network node.

5. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate access to data stored on a communications network node.

6. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate the transmission of data from a communications network node to the portable device.

7. The portable device according to claim **6**, wherein the portable device is configured to receive a data stream transmitted from a communications network node.

8. The portable device according to claim **7**, wherein the portable device is configured to receive a data stream comprising video data content.

9. The portable device according to claim **8**, wherein the terminal output component comprises a video monitor and the portable device is configured to transmit video data content to the terminal for display on the terminal video monitor.

10. The portable device according to claim **7**, wherein the portable device is configured to receive a data stream comprising audio data content.

11. The portable device according to claim **10**, wherein the terminal output component comprises a speaker and the portable device is configured to transmit audio data content to the terminal for presentation by the terminal speaker.

12. The portable device according to claim **7**, wherein the portable device is configured to receive a data stream comprising a live data feed.

13. The portable device according to claim **12**, wherein the portable device is configured to transmit to the terminal a live data stream received from a communications network node.

14. The portable device according to claim **7**, wherein the portable device is configured to transmit to the terminal the data stream received from the communications network node.

15. The portable device according to claim **6**, wherein the portable device is configured to transmit to the terminal data received from the communications network node.

16. The portable device according to claim **1**, wherein the portable device communication interface comprises an audio/video interface.

17. The portable device according to claim **1**, wherein the portable device communication interface comprises a wireless communication interface.

18. The portable device according to claim **1**, wherein the executable program code stored on the portable device memory comprises an operating system configured to be executed by the portable device processor.

19. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate the download of program code from a communications network node to the portable device.

20. The portable device according to claim **1**, wherein the portable device is configured to receive program code from a communications network node.

21. The portable device according to claim **20**, wherein the received program code comprises an updated version of the first program code and the portable device is configured to store the updated version of the first program code on the portable device memory.

22. The portable device according to claim **21**, wherein the updated version of the first program code comprises product advertisement information to be presented by the interactive user interface.

23. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate the upload of data from the portable device to a communications network node.

24. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate the upload of program code from the portable device to a communications network node.

25. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate synchronizing data stored on the portable device memory with data stored on a communications network node.

26. The portable device according to claim **1**, wherein the third program code is configured to cause a communication to be transmitted to the communications network node to facilitate synchronizing program code stored on the portable device memory with program code stored on a communications network node.

27. The portable device according to claim **1**, wherein the interactive user interface comprises a graphic user interface.

28. The portable device according to claim **1**, wherein the portable device processor is configured to execute the first program code.

29. The portable device according to claim **1**, wherein the communications network node comprises a server.

30. The portable device according to claim **1**, wherein the portable device is configured to provide the terminal processor with access to the first program code.

31. The portable device according to claim **1**, wherein the first program code is configured to be executed by the terminal processor.

32. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, a network interface configured to enable the transmission of communications between the portable device and a communications network node, and a communication interface for enabling the transmission of communications to a terminal comprising a processor and an output component, the method comprising:

(a) executing first program code stored on the portable device memory to present an interactive user interface on the terminal output component;

(b) executing second program code stored on the portable device memory to facilitate communications between the portable device and the communications network node;

(c) executing third program code stored on the portable device memory in response to user interaction with the interactive user interface; and

(d) transmitting a communication through the portable device network interface to the communications network node.

33. The method according to claim **32**, wherein the step of transmitting a communication through the portable device network interface to the communications network node facilitates verification of the portable device.

34. The method according to claim **32**, wherein the step of transmitting a communication through the portable device network interface to the communications network node facilitates the download of data to the portable device.

35. The method according to claim **32**, wherein the step of transmitting a communication through the portable device network interface to the communications network node facilitates the transmission of data from the communications network node to the portable device.

36. The method according to claim **35**, further comprising receiving data transmitted by the communications network node.

37. The method according to claim **36**, further comprising transmitting data from the portable device to the terminal.

38. The method according to claim **36**, wherein the data received by the portable device comprises a data stream.

39. The method according to claim **38**, wherein the data stream received by the portable device comprises video data content.

40. The method according to claim **39**, further comprising transmitting video data content from the portable device to the terminal for display on the terminal output component.

41. The method according to claim **38**, wherein the data stream received by the portable device comprises audio data content.

42. The method according to claim **41**, further comprising transmitting audio data content from the portable device to the terminal for presentation by the terminal output component.

43. The method according to claim **38**, wherein the data stream received by the portable device comprises a live data feed.

44. The method according to claim **43**, further comprising transmitting live data feed from the portable device to the terminal.

45. The method according to claim **32**, wherein the step of executing third program code stored on the portable device memory in response to user interaction with the interactive user interface causes the transmission of a communication through the portable device network interface to the communications network node.

46. A non-transitory computer readable medium containing executable program code to be executed by a portable device, the portable device comprising a processor, a memory, a network interface for enabling communications between the portable device a communications network node, and a communication interface for enabling the transmission of communications from the portable device to a terminal comprising a processor and an output component, the program code comprising:

(a) first program code which, when executed, is configured to present an interactive user interface on the terminal output component,

(b) second program code which, when executed by the portable device processor, is configured to facilitate communications between the portable device and the communications network node; and

(c) third program code which, when executed by the portable device processor in response to user interaction with the interactive user interface, is configured to cause a communication to be transmitted through the portable device network interface to the communications network node.

47. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, a network interface configured to enable the transmission of communications between the portable device and a communications network node, and a communication interface for enabling the transmission of communications to a terminal comprising a processor and an output component, the method comprising:

(a) causing an interactive user interface to be presented by the terminal output component;

(b) executing first program code stored on the portable device memory to facilitate communications between the portable device and the communications network node;

(c) executing second program code stored on the portable device memory in response to user interaction with the interactive user interface; and

(d) transmitting a communication through the portable device network interface to the communications network node.

48. The method according to claim **47**, wherein the step of executing second program code stored on the portable device memory in response to user interaction with the interactive user interface causes the transmission of a communication through the portable device network interface to the communications network node.

49. The method according to claim **47**, wherein the step of causing an interactive user interface to be presented by the terminal output component comprises providing the terminal with access to third program code stored on the portable device memory which, when executed by the terminal processor, is configured to cause an interactive user interface to be presented by the terminal output component.

50. The method according to claim **47**, wherein the step of causing an interactive user interface to be presented by the terminal output component comprises executing third program code stored on the portable device memory cause an interactive user interface to be presented by the terminal output component.

51. The method according to claim **50**, wherein the portable device executes the third program code in response to user interaction with an interactive user interface on the terminal output component.

52. The method according to claim **47**, wherein the step of causing an interactive user interface to be presented by the terminal output component comprises affecting the presentation of an interactive user interface on the terminal output component.

53. The method according to claim **52**, wherein the portable device executes third program code stored on the portable device memory to affect the presentation of the interactive user interface on the terminal output component.

54. The method according to claim **53**, where the portable device executes the third program code in response to user interaction with the interactive user interface on the terminal output component.

55. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, and an external communication interface for enabling the transmission of a plurality of communications between the portable device and a terminal, the terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the method comprising:

(a) causing the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component;

(b) executing third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal network communication interface;

(c) executing, in response to a communication received by the portable device resulting from user interaction with the interactive user interface, fourth program code stored on the portable device memory to cause a communication to be transmitted to a communications network node; and

(d) facilitating communications through the terminal network communication interface to a communications network node.

56. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate verification of the portable device.

57. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the transmission of encrypted communications from the communications network node to the terminal.

58. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate access to the communications network node.

59. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the download of content from the communications network node to the terminal.

60. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the download of content from the communications network node to the portable device.

61. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the download of program code from the communications network node to the terminal.

62. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the download of program code from the communications network node to the portable device.

63. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the upload of content to the communications network node.

64. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the upload of program code to the communications network node.

65. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate synchronizing content on the portable device with content on the communications network node.

66. The method according to claim **55**, wherein the step of executing fourth program code stored on the portable device memory causes a communication to be transmitted to the communications network node to facilitate the transmission of a live data feed to the terminal.

67. The method according to claim **55**, wherein the step of causing the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component comprises transmitting a communication to the terminal.

68. The method according to claim **55**, wherein the step of causing the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component comprises executing fifth program code stored on the portable device memory to transmit a communication to cause the terminal to execute the first program code.

69. The method according to claim **68**, wherein the portable device executes the fifth program code in response to user interaction with an interactive user interface presented by the terminal output component.

70. The method according to claim **55**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises transmitting a communication to the terminal to cause the communication to the communications network node.

71. The method according to claim **55**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with data stored on the portable device memory to facilitate the terminal to transmit a communication to the communications network node.

72. The method according to claim **71**, wherein the data stored on the portable device memory comprises user authorization data.

73. The method according to claim **72**, wherein the user authorization information is user biometric data.

74. The method according to claim **71**, wherein the data stored on the portable device memory comprises a digital certificate.

75. The method according to claim **71**, wherein the data stored on the portable device memory comprises portable device identifier information.

76. The method according to claim **55**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with biometric data to facilitate the terminal to transmit a communication to the communications network node.

77. The method according to claim **55**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises effecting the execution of fifth program code to facilitate the communication to the communications network node.

78. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, and an external communication interface for enabling the transmission of a plurality of communications between the portable device and a terminal, the terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the method comprising:

(a) affecting the presentation of an interactive user interface by the terminal output component;

(b) executing second program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal network communication interface;

(c) executing, in response to a communication received by the portable device resulting from user interaction with the interactive user interface, third program code stored on the portable device memory to cause a communication to be transmitted to a communications network node; and

(d) facilitating communications through the terminal network communication interface to a communications network node.

79. The method according to claim **78**, wherein the step of affecting the presentation of an interactive user interface by the terminal output component comprises providing the terminal processor with access to fourth program code stored on the portable device memory which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component.

80. The method according to claim **79**, wherein the step of providing the terminal processor with access to the fourth program code stored on the portable device memory which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component comprises storing the fourth program code onto the terminal memory.

81. The method according to claim **78**, wherein the step of affecting the presentation of an interactive user interface by the terminal output component comprises executing fourth program code stored on the portable device memory to affect the presentation of an interactive user interface by the terminal output component.

82. The method according to claim **78**, wherein the step of affecting the presentation of an interactive user interface by the terminal output component comprises transmitting a communication to cause the terminal processor to execute fourth program code stored on the terminal memory to affect the presentation of an interactive user interface by the terminal output component.

83. The method according to claim **78**, wherein the step of affecting the presentation of an interactive user interface by the terminal output component comprises executing fifth program code stored on the portable device memory to transmit a communication to cause the terminal to execute fourth program code stored on the terminal memory to affect the presentation of an interactive user interface by the terminal output component.

84. The method according to claim **83**, wherein the portable device executes the fifth program code in response to user interactive with an interactive user interface presented by the terminal output component.

85. The method according to claim **78**, wherein the step of executing third program code to cause a communication to be transmitted to a communications network node comprises transmitting a communication to the terminal to cause the communication to the communications network node.

86. The method according to claim **78**, wherein the step of executing third program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with data stored on the portable device memory to facilitate the terminal to transmit a communication to the communications network node.

87. The method according to claim **86**, wherein the data stored on the portable device memory comprises user authorization data.

88. The method according to claim **87**, wherein the user authorization information is user biometric data.

89. The method according to claim **86**, wherein the data stored on the portable device memory comprises a digital certificate.

90. The method according to claim **86**, wherein the data stored on the portable device memory comprises portable device identifier information.

91. The method according to claim **78**, wherein the step of executing third program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with biometric data to facilitate the terminal to transmit a communication to the communications network node.

92. The method according to claim **78**, wherein the step of executing third program code to cause a communication to be transmitted to a communications network node comprises effecting the execution of fifth program code to facilitate the communication to the communications network node.

93. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, and an external communication interface for enabling the transmission of a plurality of communications between the portable device and a terminal, the terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the method comprising:

(a) affecting the presentation of the interactive user interface presented by the terminal output component;

(b) executing third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal network communication interface;

(c) executing, in response to a communication received by the portable device resulting from user interaction with the interactive user interface, fourth program code stored on the portable device memory to cause a communication to be transmitted to a communications network node; and

(d) facilitating communications through the terminal network communication interface to a communications network node.

94. The method according to claim **93**, wherein the portable device executes fifth program code stored on the portable device memory to affect the presentation of the interactive user interface by the terminal output component.

95. The method according to claim **94**, wherein the portable device executes the fifth program code in response to user interaction with the interactive user interface presented by the terminal output component.

96. The method according to claim **93**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises transmitting a communication to the terminal to cause the communication to the communications network node.

97. The method according to claim **93**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with data stored on the portable device memory to facilitate the terminal to transmit a communication to the communications network node.

98. The method according to claim **97**, wherein the data stored on the portable device memory comprises user authorization data.

99. The method according to claim **98**, wherein the user authorization information is user biometric data.

100. The method according to claim **97**, wherein the data stored on the portable device memory comprises a digital certificate.

101. The method according to claim **97**, wherein the data stored on the portable device memory comprises portable device identifier information.

102. The method according to claim **93**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises providing the terminal with biometric data to facilitate the terminal to transmit a communication to the communications network node.

103. The method according to claim **93**, wherein the step of executing fourth program code to cause a communication to be transmitted to a communications network node comprises effecting the execution of fifth program code to facilitate the communication to the communications network node.

104. A system implementing a terminal having a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to affect the presentation of an interactive user interface by the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to and from the terminal, the system comprising:

(a) a communications network node; and

(b) a portable device comprising an external communication interface for enabling the transmission of a plurality of communications between the portable device and the terminal, a processor, and a memory, wherein the memory has executable program code stored thereon, the portable device configured to:

(1) cause the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component;

(2) execute third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal network communication interface;

(3) execute fourth program code stored on the portable device memory in response to a communication received by the portable device resulting from user interaction with the interactive user interface to cause a communication to be transmitted to a communications network node; and

(4) facilitate communications through the terminal network communication interface to a communications network node.

105. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate verification of the portable device.

106. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the transmission of encrypted communications from the communications network node to the terminal.

107. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate access to the communications network node.

108. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the download of content from the communications network node to the terminal.

109. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the download of content from the communications network node to the portable device.

110. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the download of program code from the communications network node to the terminal.

111. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the download of program code from the communications network node to the portable device.

112. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the upload of content to the communications network node.

113. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the upload of program code to the communications network node.

114. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate synchronizing content on the portable device with content on the communications network node.

115. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to cause a communication to be transmitted to the communications network node to facilitate the transmission of a live data feed to the terminal.

116. The system according to claim **104**, wherein the portable device is configured to transmit a communication to the terminal to cause the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component.

117. The system according to claim **104**, wherein the portable device is configured to execute fifth program code stored on the portable device memory to cause the terminal to execute the first program code to affect the presentation of an interactive user interface by the terminal output component.

118. The system according to claim **117**, wherein the portable device is configured to execute the fifth program code in response to user interaction with an interactive user interface presented by the terminal output component.

119. The system according to claim **104**, wherein the portable device is configured execute the fourth program code to transmit a communication to the terminal to cause the communication to the communications network node.

120. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to provide the terminal with data stored on the portable device to facilitate the terminal to transmit a communication to the communications network node.

121. The system according to claim **120**, wherein the data stored on the portable device memory comprises user authorization data.

122. The system according to claim **121**, wherein the user authorization information comprises user biometric data.

123. The system according to claim **120**, wherein the data stored on the portable device memory comprises a digital certificate.

124. The system according to claim **120**, wherein the data stored on the portable device memory comprises portable device identifier information.

125. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to provide the terminal with biometric data to facilitate the terminal to transmit a communication to the communications network node.

126. The system according to claim **104**, wherein the portable device is configured to execute the fourth program to effect the execution of fifth program code to cause a communication to be transmitted to the communications network node.

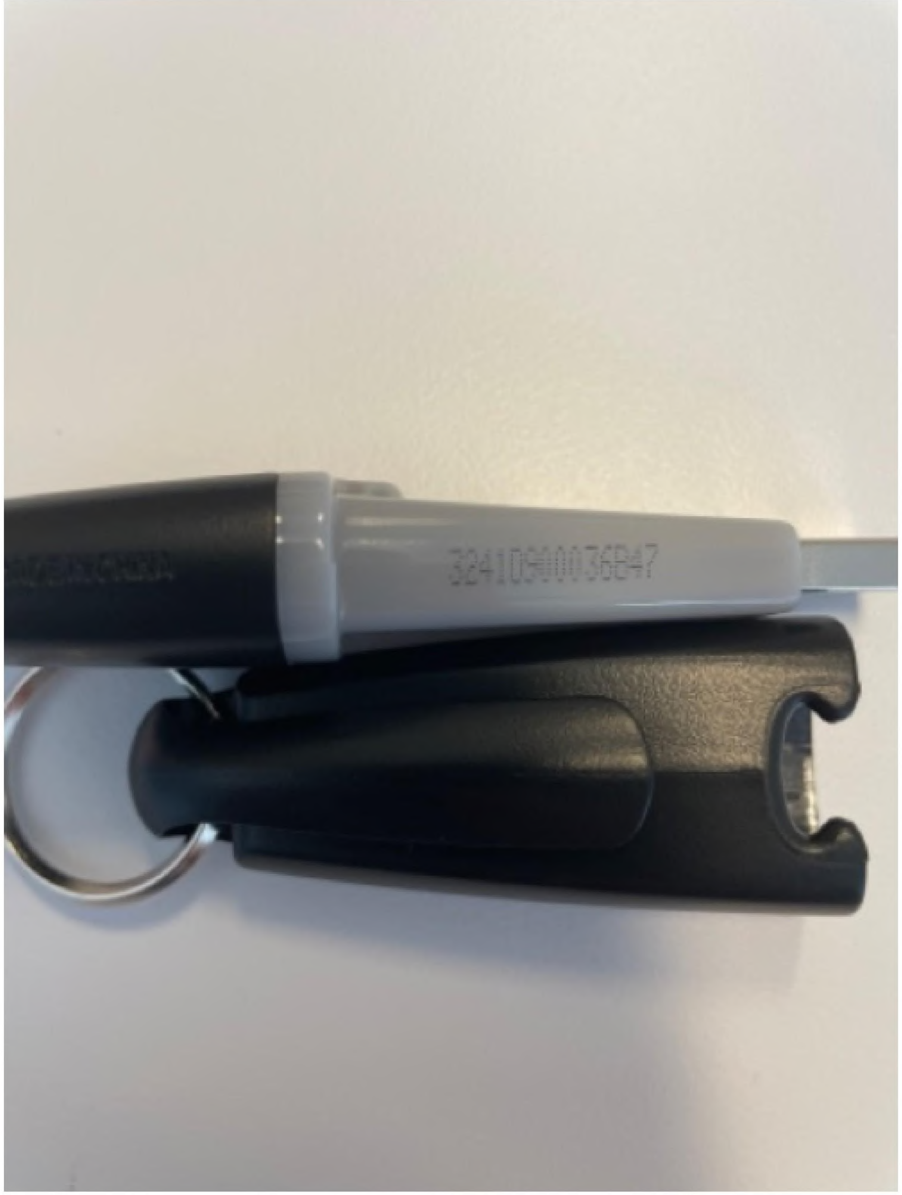
127. The system according to claim **126**, wherein the portable device is configured to execute the fifth program code.

128. The system according to claim **127**, wherein the portable device is configured to execute the fifth program code in response to user interaction with the interactive user interface presented by the terminal output component.

129. The system according to claim **104**, wherein the portable device is configured to execute the fourth program code to effect the execution of fifth program code to facilitate the communication to the communications network node.

* * * * *

EXHIBIT 3



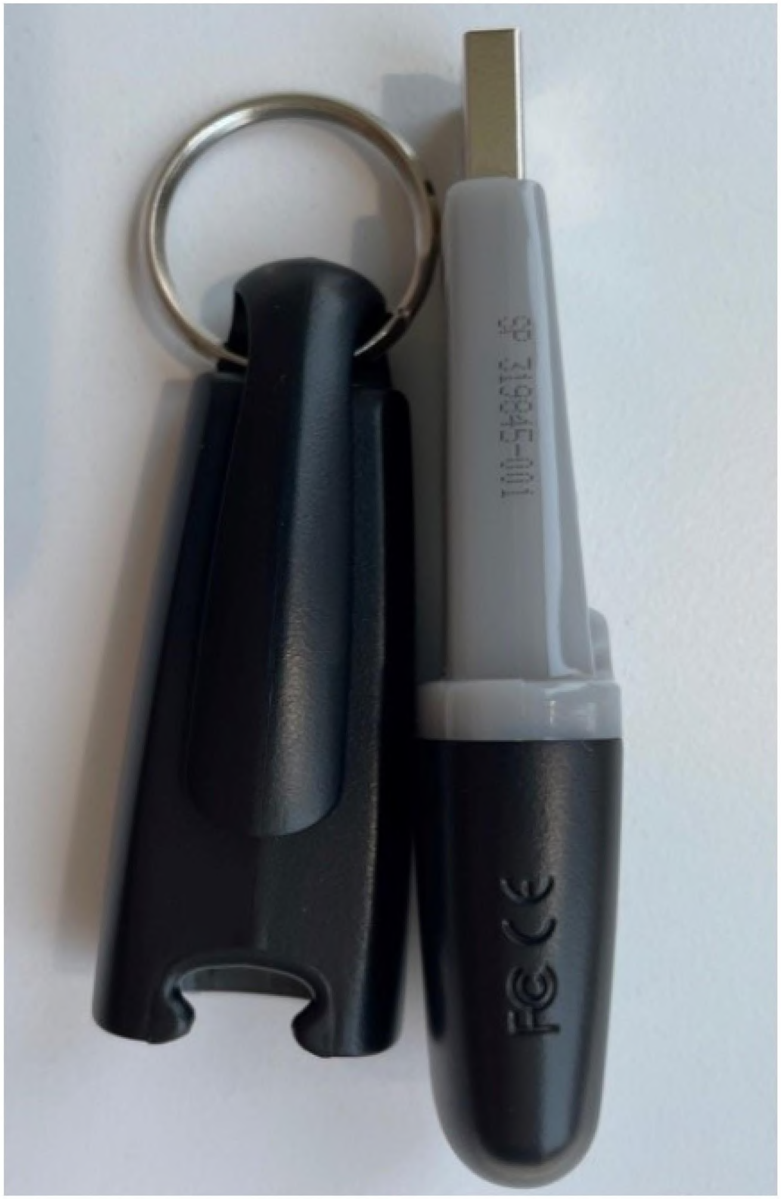
Ingenico Inc. v. IOENGINE LLC

DX-170

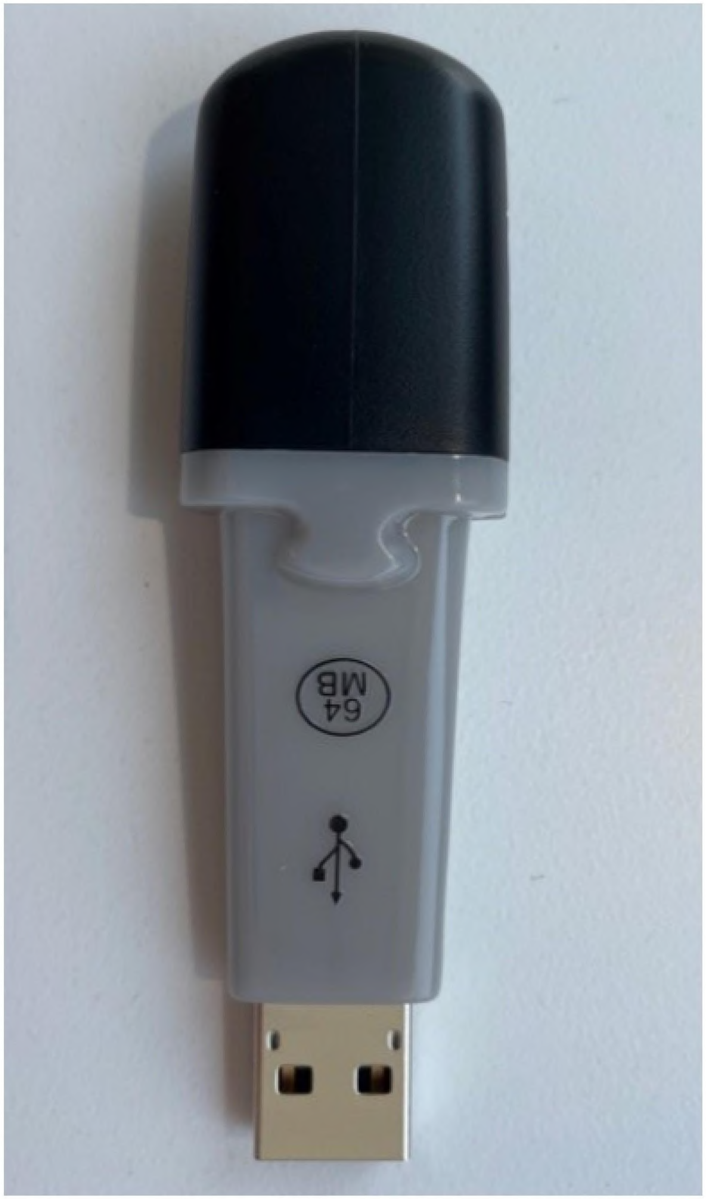
C.A. No. 18-cv-00826-WCB

Exhibit 0004

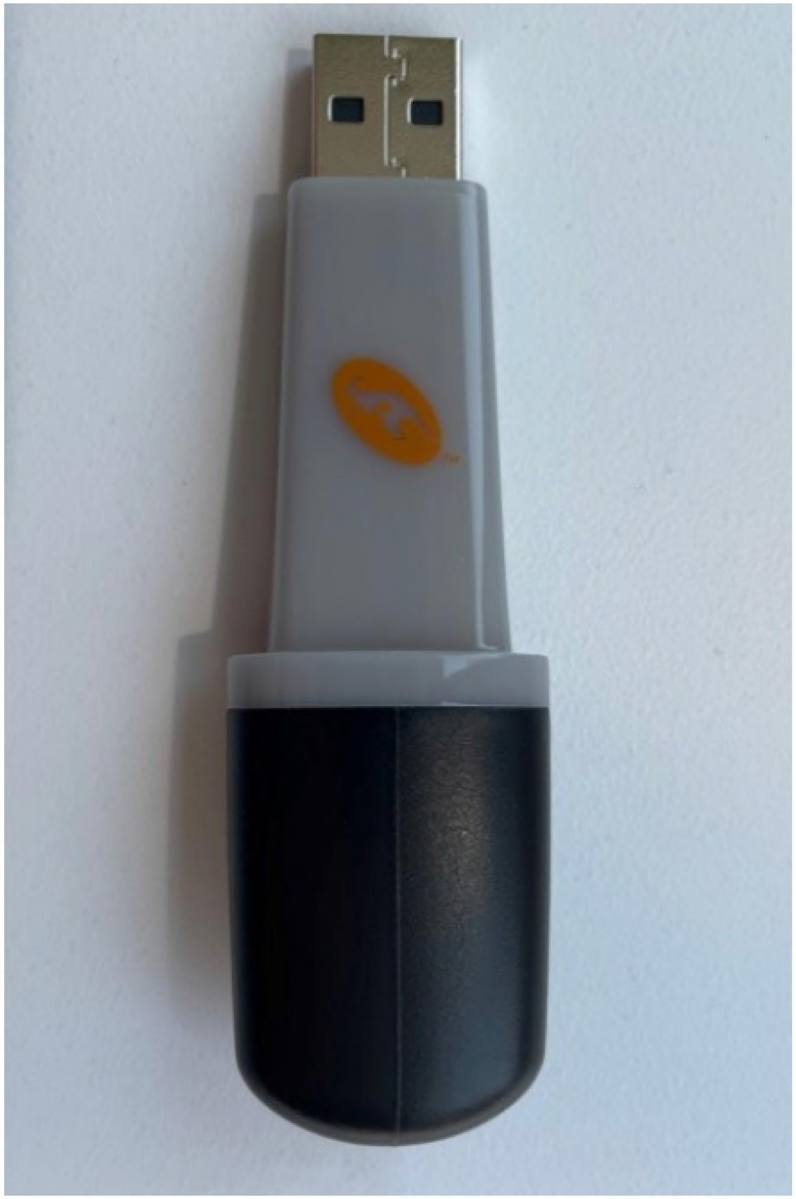
Sobol



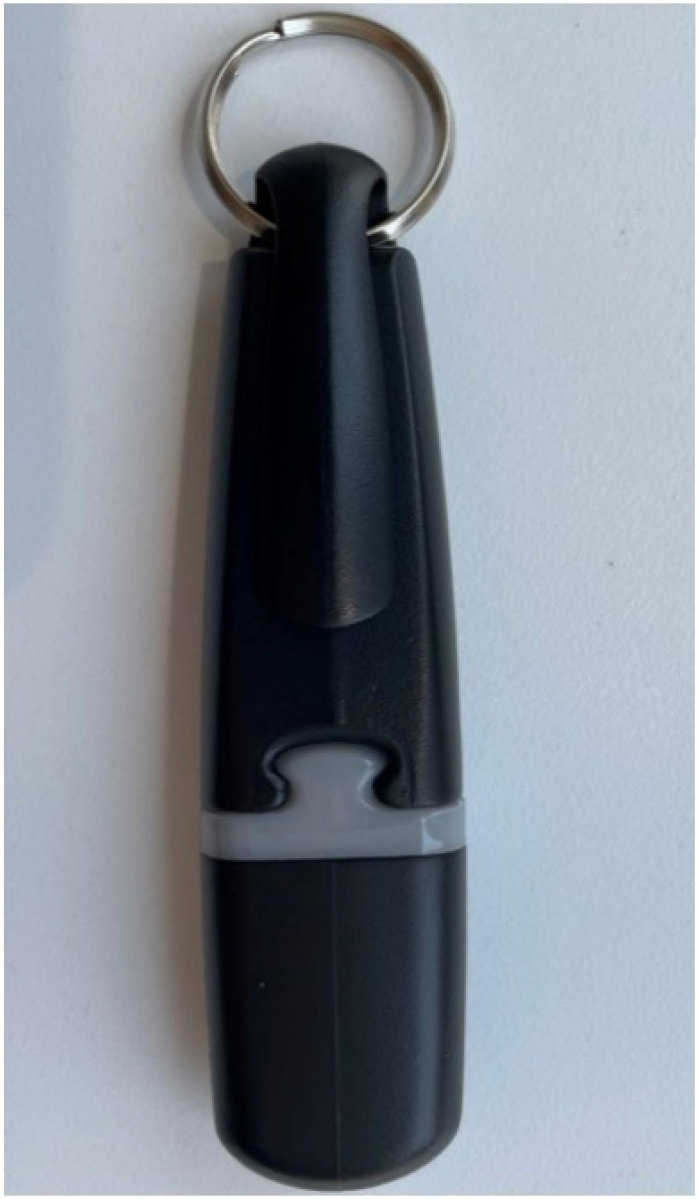
DX-170.2



DX-170.3



DX-170.4



DX-170.5

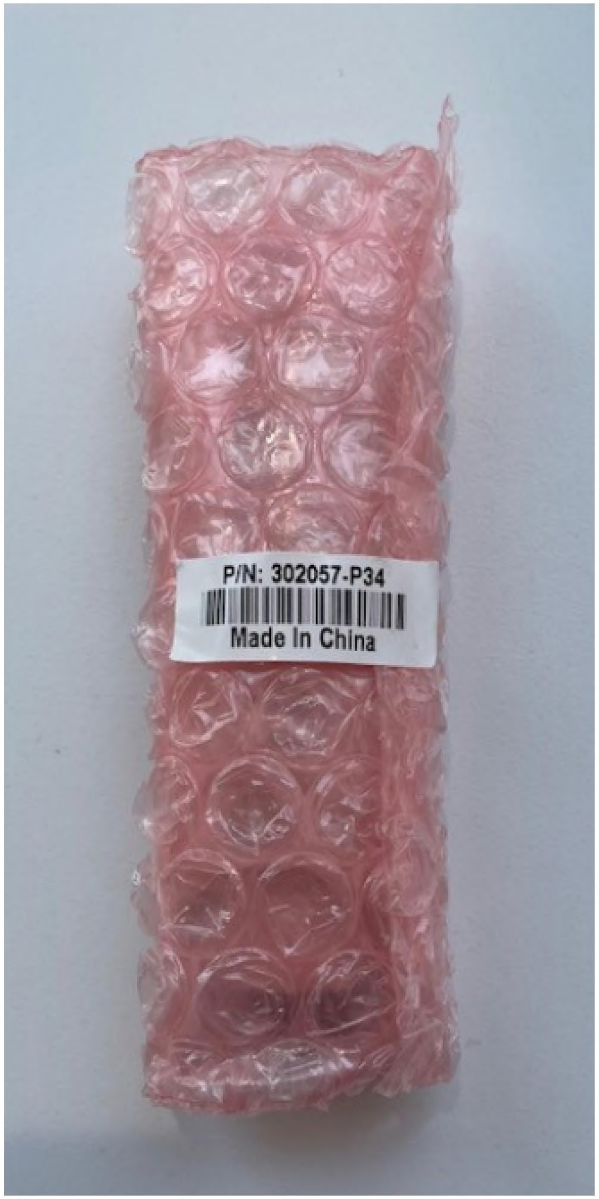

P/N: 302057-P34
Made In China

EXHIBIT 4

DISKonKEY®

M-Systems
Flash Disk Pioneers

Using DiskOnKey Upgrade

DiskOnKey can be much more than an easy-to-use removable storage device operating through your USB port. DiskOnKey Upgrade allows you to upgrade your DiskOnKey remotely with the most up-to-date software version to support a complete range of customization and enhanced functionality applications, such as KeySafe to protect your data. This readme file details the procedures for using DiskOnKey Upgrade.

1 System Requirements

- Pentium II 266MHz
- 600KB free disk space
- Operating Systems: Windows 98®[1]SE, Windows ME®, Windows 2000®, Windows XP®
- Administrative privileges for Windows 2000 and XP
- An Internet connection (minimum speed: 28,800 bps)

2 Downloading The Application

No installation is required. Simply download the latest version of the DiskOnKey Upgrade from http://www.diskonkey.com/ and save it to your hard disk.

3 Running the Application

Before you begin running the DiskOnKey Upgrade application, make sure you have an active connection to the Internet. Then follow the steps below:

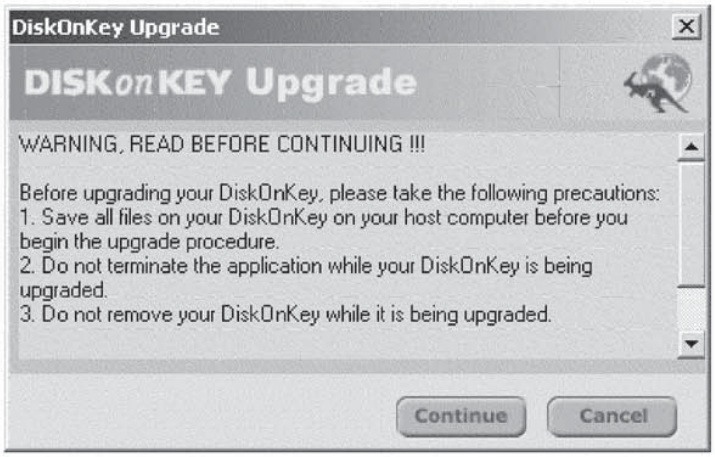
1. Double click the DiskOnKey Upgrade application icon.
2. The first screen contains instructions for using the application, both before and during the upgrading procedure.

   IMPORTANT! To prevent data loss, it is essential to follow these instructions.
3. Click 'Continue' to advance to the next screen, or 'Cancel' to exit the application (Figure 1: Opening Screen).

[1] Windows 98 requires a driver that you can download at http://www.diskonkey.com/. Make sure you have the latest DiskOnKey driver installed before using the upgrade application.

Ingenico Inc. v. IOENGINE LLC
DX-331
C.A. No. 18-cv-00826-WCB

Exhibit
0021
Sobol



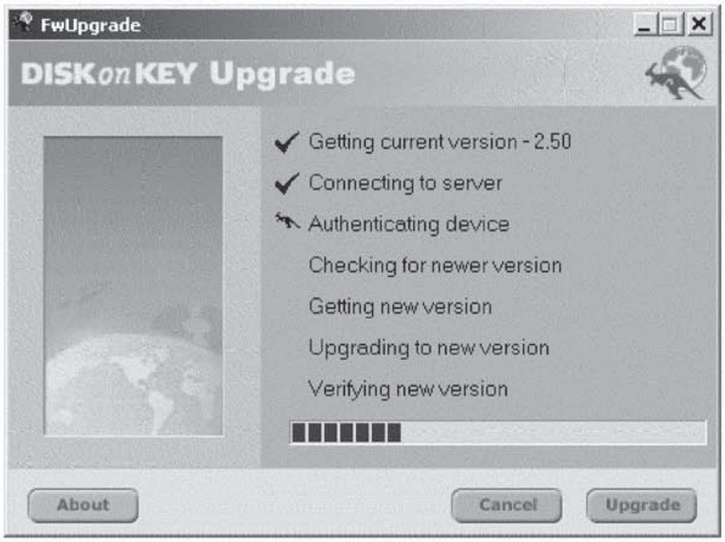
WARNING, READ BEFORE CONTINUING !!!

Before upgrading your DiskOnKey, please take the following precautions:
1. Save all files on your DiskOnKey on your host computer before you begin the upgrade procedure.
2. Do not terminate the application while your DiskOnKey is being upgraded.
3. Do not remove your DiskOnKey while it is being upgraded.

FIGURE 1: OPENING SCREEN

4. Click 'Upgrade' to begin the upgrade procedure. This button is enabled only if your DiskOnKey is connected. The application connects to a remote server and checks if a newer version exists for the current DiskOnKey. The steps of this process are displayed on screen (Figure 2: Upgrade Screen).

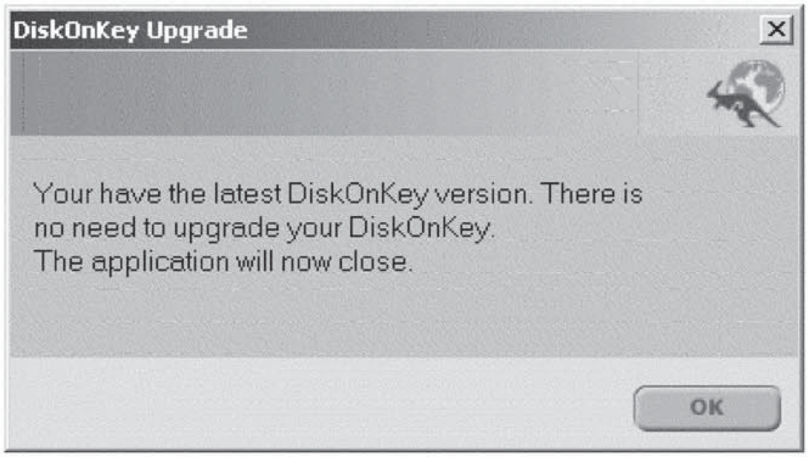
   Note: If this is first time that you are inserting your DiskOnKey in the computer, wait until the registration procedure finishes before beginning to use it.



✓ Getting current version - 2.50
✓ Connecting to server
Authenticating device
Checking for newer version
Getting new version
Upgrading to new version
Verifying new version

FIGURE 2: UPGRADE SCREEN

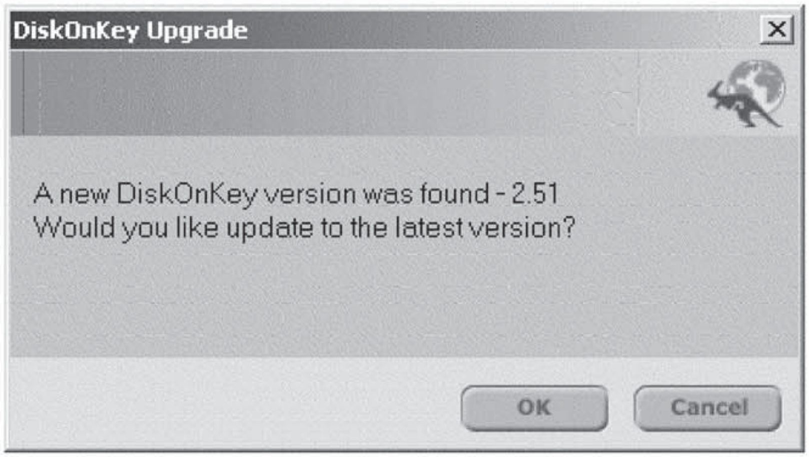
5. If your DiskOnKey is already using the newest version, click 'OK' to end the application after the screen below is displayed (Figure 3: No New Version Exists).



Your have the latest DiskOnKey version. There is no need to upgrade your DiskOnKey.
The application will now close.

FIGURE 3: NO NEW VERSION EXISTS

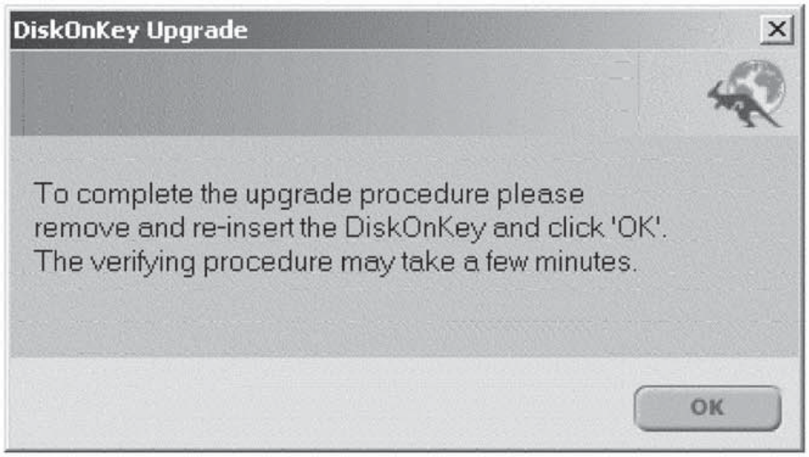
6. If not, click 'OK' to upgrade to the new version or 'Cancel' to end the application (Figure 4: New Version Found).



A new DiskOnKey version was found - 2.51
Would you like update to the latest version?

FIGURE 4: NEW VERSION FOUND

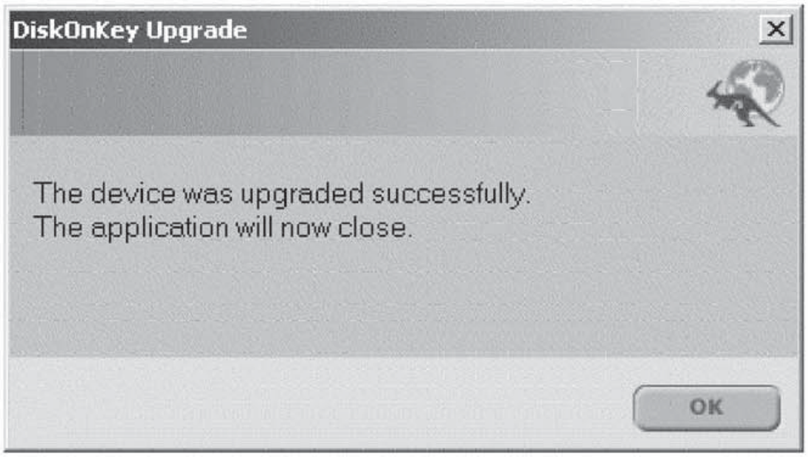
7. To complete the upgrade procedure, remove and re-insert your DiskOnKey, which is similar in action to restarting your computer (Figure 5: Reinsert DiskOnKey).



To complete the upgrade procedure please remove and re-insert the DiskOnKey and click 'OK'. The verifying procedure may take a few minutes.

FIGURE 5: REINSERT DISKONKEY

8. If the upgrade procedure was successful, click 'OK' to end the application after the screen below is displayed (Figure 6: Upgrade Successful).



The device was upgraded successfully.
The application will now close.

FIGURE 6: UPGRADE SUCCESSFUL

4 TROUBLESHOOTING

4.1 Multiple DiskOnKeys Inserted

This application can update one DiskOnKey at a time. If you have attempted to upgrade multiple DiskOnKeys simultaneously, remove all but one before attempting to perform the upgrade.

4.2 Privacy Zone Detected

If your DiskOnKey has a privacy zone (a DiskOnKey KeySafe feature), remove it before trying to upgrade the software.

4.3 No Communication with Server

If the application is unable to communicate with the server, check your Internet connection and try again.

4.4 Current Version Cannot be Extracted

If the application cannot extract the current version of your DiskOnKey, re-insert it and try again.

4.5 Current Version No Longer Supported

If the version installed on your DiskOnKey is no longer valid, download the latest version from the DiskOnKey website (*www.diskonkey.com*).

4.6 Upgrade Procedure Failed

If you receive an upgrade procedure failure message, perform the upgrade procedure again, as described in this document. If it succeeds on the next attempt, the message “DiskOnKey is already using the newest version. Click ‘OK’ to terminate the application” will be displayed.

Bates Beg :	WDC0008108
Bates End :	WDC0008112
Confidential Designation :	Confidential - Attorneys' Eyes Only
File Type :	Adobe Portable Document Format
Date and Time Last Modified :	7/8/2002 6:03:00 AM
Date and Time Created :	7/8/2002 7:03:00 AM
Email From :	
Email To :	
Email CC :	
Email BCC :	

DX-331.6

EXHIBITS 5-6
REDACTED IN THEIR ENTIRETY

EXHIBIT 7

FILED UNDER SEAL

CONFIDENTIAL – ATTORNEYS' EYES ONLY
CONFIDENTIAL – OUTSIDE ATTORNEYS' EYES ONLY – SOURCE CODE

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IOENGINE, LLC, Plaintiff, v. PAYPAL HOLDINGS, INC., Defendant.	C.A. No. 1:18-cv-452 GMS Jury Trial Demanded

PAYPAL'S SUPPLEMENTAL RESPONSES AND OBJECTIONS TO IOENGINE'S INTERROGATORY NOS. 1-4, 6-7, 9, 11-13, 15-19, AND 22-23

Pursuant to Federal Rules of Civil Procedure 26 and 33, Defendant PayPal Holdings, Inc. ("Defendant" or "PayPal") hereby objects and responds to Plaintiff IOENGINE, LLC's ("Plaintiff's" or "IOENGINE's") Interrogatory Nos. 1-4, 6-7, 9, 11-13, 15-19, and 22-23.

PRELIMINARY STATEMENT

PayPal has made a reasonable investigation for information responsive to IOENGINE's interrogatories based upon PayPal's current knowledge, information, and belief. PayPal's investigation is ongoing. PayPal's responses are made without prejudice to its right to revise, correct, supplement, or clarify its responses at any time pursuant to Federal Rule of Civil Procedure 26(e). PayPal reserves the right to make any use of, or to introduce at any hearing or trial, information responsive to these interrogatories that was discovered after the date of this response.

By providing responses, PayPal does not concede the relevancy of the subject matter of any interrogatory. PayPal reserves all objections or other questions as to the competency, relevance, materiality, privilege, or admissibility in any proceeding or trial for any purpose whatsoever of its responses provided herein. PayPal provides these written responses subject to the general and specific objections stated below, and subject to PayPal's right to object at any proceeding involving or relating to the subject matter of the requests responded to herein.

By providing responses, PayPal does not waive any of its rights and limitations on discovery set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, any other applicable law, rule, or order, and all other existing or future agreements of the parties concerning these proceedings, including discovery.

PayPal's responses contained herein and the information produced are provided solely for the purpose of discovery in this action. PayPal's responses are not to be used for any purpose outside the context of this lawsuit.

GENERAL OBJECTIONS

PayPal's responses to these interrogatories are subject to the following General Objections, which are hereby incorporated into PayPal's responses to the requests provided further below.

1. PayPal objects to these interrogatories to the extent they seek disclosure of information protected from disclosure under the attorney-client privilege, the doctrine of work product immunity, the common interest and joint defense privileges, or any other claim of privilege, law or rule ("Privileged Information"). PayPal will not disclose Privileged Information in response to these interrogatories, and PayPal's undertaking to respond should be understood to exclude Privileged Information. Any disclosure of Privileged Information is inadvertent and should not be construed to constitute a waiver. If PayPal withholds Privileged Information, it will log that information and respond in accordance with the terms of a suitable agreement to be negotiated pursuant to Delaware Default Standard for Discovery Paragraph 1.d.

2. PayPal objects to these interrogatories to the extent they seek disclosure of information protected by the rights of privacy of third parties or to the extent they seek disclosure of confidential information of third parties to which PayPal owes an obligation of confidentiality or duty to prevent disclosure, including but not limited to information subject to confidentiality agreements, non-disclosure agreements, protective orders, or the General Data Protection Regulation in the European Union.

3. PayPal objects to these interrogatories to the extent they seek to impose obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. For example, PayPal objects to these requests to the extent they are overly broad, unduly burdensome, harassing, and/or oppressive and to the extent they seek information or documents neither relevant to a claim or defense of any party nor proportional to the needs of the case, considering the importance of the issues at stake in the action, the amount in controversy, the parties' relative access to relevant information, the parties' resources, the importance of the discovery in resolving the issues, and whether the burden or expense of the proposed discovery outweighs its likely benefit.

4. PayPal objects to these interrogatories to the extent they seek information or documents not in its possession, custody, or control or purport to require PayPal to respond on behalf of another entity or call for information not within PayPal's own knowledge.

5. PayPal objects to these interrogatories to the extent they purport to shift the burden of proving any fact or issue borne by IOENGINE onto PayPal.

6. PayPal objects to these interrogatories to the extent they seek publicly available information, information already within IOENGINE's possession, custody, or control, or information that IOENGINE could obtain as easily as could PayPal.

7. PayPal objects to these interrogatories to the extent they attempt to impose a duty on PayPal to undertake a search for information beyond a reasonable search of PayPal's files where PayPal would expect to find information responsive to the requests.

GENERAL OBJECTIONS TO "DEFINITIONS" AND "INSTRUCTIONS"

1. In addition to the General Objections above, PayPal's responses to these interrogatories are subject to the following objections to the "Definitions" and "Instructions" provided with the requests, each of which are hereby incorporated by reference into PayPal's responses below. If PayPal objects to a definition of a term and that term is used in the definition of a subsequent term, PayPal's objections to the term used in the subsequent definition are incorporated by reference therein.

2. PayPal objects to IOENGINE's "Definitions" and "Instructions" to the extent that they contradict and/or impose obligations upon PayPal in excess of or different from the duties and obligations imposed by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

3. PayPal objects to the definition of the terms "PayPal," "Zettle," "You," and "Your" on the ground that they are overly broad and unduly burdensome, seek information that is privileged or immune from discovery, and include persons or entities that PayPal has no control over and, therefore, PayPal does not have possession, custody, or control of information of any such persons or entities. PayPal will interpret each of these terms—"PayPal," "Zettle," "You," and "Your," and similar terms—to mean PayPal Holdings, Inc.

4. PayPal objects to the definition of the terms "Document" and "All Documents" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definitions of the terms "Document" and "All Documents" to the extent they call for the production of email and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

5. PayPal objects to the definition of the term "Communication" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definition of the term "Communication" to the extent it calls for the production of e-mail, Internet communications, and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

6. PayPal objects to the definitions of the terms "Thing," "Relate," "Relate To," "Relating To," "Concern," "Concerning," and "Person" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order.

7. PayPal objects to the definition of the terms "Identify" and "Describe" as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent

identification or description is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

8. PayPal objects to the definitions of the terms "Accused mPOS Product," "Accused PayPal Product(s)," "Ingenico-Supplied PayPal Products," "Accused PayPal Devices," "PayPal Here App," "PayPal POS Partner App," "Accused Mobile Payment Apps," and "PayPal Here Service," as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they purport to relate to products or services other than those specifically identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret these terms to mean only the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

9. PayPal objects to the definitions of the terms "Zettle Service" and "Zettle App" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret these terms to mean only the Zettle App (*see* https://play.google.com/store/apps/details?id=com.izettle.android&hl=en_US&gl=US and https://apps.apple.com/us/app/paypal-zettle-point-of-sale/id447785763) and Zettle Reader as released in the United States (*see* www.paypal.com/us/business/pos/payments/card-reader).

10. PayPal objects to the definition of the term "Transaction Revenue" as overly broad, unduly burdensome, not proportional to the needs of the case, vague, and ambiguous to the extent it purports to include fees paid "for the benefit of PayPal" and all fees "Relating To any Accused PayPal Products." PayPal further objects to IOENGINE's request for transaction revenue as not relevant to any party's claim or defense and not proportional to the needs of the

case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention.

11. PayPal objects to the definition of the term "Accused Functionality" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it purports to relate to functionalities other than those that are identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret this term to mean those functionalities.

12. PayPal objects to the definition of the terms "POS Partner App," "POS Partner," and "POS Partners" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret the term "POS Partner App" to mean only the Lavu Point of Sale App, TouchBistro App, Vend Register App, ERPLY cloud-based POS software, Brightpearl POS system, and TouchPoint POS system after June 2015. PayPal will interpret the terms "POS Partner" and "POS Partners" to mean only the providers of the aforementioned POS Partner Apps.

13. PayPal objects to the definition of the term "Mobile Credit Card Processing Market" as vague, overly broad, unduly burdensome, and not proportional to the needs of the case, as it merely points to ten pages of a hearing transcript that did not "ascribe" any specific meaning to the term, as IOENGINE incorrectly states. As the Court clearly stated to Ingenico's counsel with respect to this issue, "[a]ll you can be expected to do is report what you have and you are not expected to go out and produce information you don't have." 2021-03-31 Hearing Tr. at 75:6-8.

14. PayPal objects to the definition of the term "Compatible Terminal" to the extent that it calls for a legal conclusion and/or improperly suggests that PayPal technology is used in

conjunction with any device that infringes any claim of any Patent-in-Suit. PayPal specifically denies any such alleged infringement. PayPal further objects to the definition of this term to be extent it suggests that the rights afforded by the Asserted Patents extend beyond the subject matter defined in the claims. PayPal will not provide responses that relate to the structure or function of any device that is not provided by PayPal, except for the operation of the "PayPal Here App" and any properly accused "PayPal POS Partner Apps," as those terms were objected to above.

15. PayPal objects to "Instructions" Paragraphs 3-11 to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

16. PayPal further objects to objects to the identifying information and descriptions requested in "Instructions" Paragraphs 3-11 as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent any of the identifying information or descriptions requested is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

17. PayPal objects to "Instructions" Paragraph 12 as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it suggests that all information relating to the time period from six years before the filing of this Action to the present is relevant, no matter what issues it relates to. PayPal will respond with respect to the time period that is relevant to each request.

GENERAL FINANCIAL INFORMATION OBJECTION

1. IOENGINE's requests and interrogatories regarding financial information are overbroad, unduly burdensome, and not proportional to the needs of the case to the extent they seek financial information other than:

- information sufficient to show quarterly revenue and costs of goods sold for U.S. sales of the products identified in PayPal's response to Interrogatory No. 1 since June 2015;
- information sufficient to show the number of units sold on a quarterly basis in the U.S. for the products identified in PayPal's response to Interrogatory No. 1 since June 2015; and
- PayPal's annual Form 10-K filings since 2015.

2. PayPal objects to IOENGINE's requests and interrogatories requesting revenue information related to providing any services, including transaction revenue, as not relevant to any party's claim or defense and not proportional to the needs of the case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention. PayPal objects to any requests or interrogatories relating to PayPal's financial information as overbroad, unduly burdensome, and not proportional to the needs of the case, to the extent they require PayPal to provide information or produce material beyond that in the bullet-point list, *supra*.

SPECIFIC OBJECTIONS AND RESPONSES

[redacted]

[redacted]

INTERROGATORY NO. 9:

Identify, including but not limited to the name and most recent contact information, any Third Parties who have created for, on behalf of, at the direction of, or with the assistance of PayPal, or otherwise provided to PayPal, any code, either executable code, source code, software development kit, or firmware code, for use on, in, by, or for the benefit of any Accused mPOS Product, and describe in detail the purpose and operation of such code.

RESPONSE TO INTERROGATORY NO. 9:

PayPal objects to this Interrogatory on the grounds that it is vague, ambiguous, and overly broad, particularly with respect to its usage of the phrases "for, on behalf of, at the direction of, or with the assistance of PayPal," "otherwise provided to PayPal," "for the benefit of," and "the purpose and operation of such code." PayPal objects to the phrase "Accused mPOS Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions."

IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App. PayPal further objects to this Interrogatory as seeking publicly available information that IOENGINE could obtain as easily as could PayPal.

PayPal further objects to this Interrogatory as seeking information that is beyond the scope permitted under Federal Rules of Civil Procedure 26(b) and 34 because, as set forth in PayPal's motion to dismiss the complaint, IOENGINE has not properly pled any claim that puts the development of third-party applications at issue, and thus it is not entitled to discovery on this issue.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds that the entities listed in Section (i) of its July 30, 2018 Fed. R. Civ. P. 26(A)(1) Initial Disclosures and Section (iii) of its August 20, 2018 Delaware Default Standard for Discovery Paragraph 3 Disclosures are likely to have knowledge or information relevant to the subject matter of the instant litigation regarding the subject matter area(s) listed for that entity.

<u>FIRST SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 9:</u>

PayPal objects to this Interrogatory on the grounds that it is vague, ambiguous, and overly broad, particularly with respect to its usage of the phrases "for, on behalf of, at the direction of, or with the assistance of PayPal," "otherwise provided to PayPal," "for the benefit of," and "the purpose and operation of such code." PayPal objects to the phrase "Accused mPOS Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions."

IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

PayPal further objects to this Interrogatory as seeking publicly available information that IOENGINE could obtain as easily as could PayPal.

PayPal further objects to this Interrogatory as seeking information that is beyond the scope permitted under Federal Rules of Civil Procedure 26(b) and 34 because, as set forth in PayPal's motion to dismiss the complaint, IOENGINE has not properly pled any claim that puts the development of third-party applications at issue, and thus it is not entitled to discovery on this issue.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004400-PP0004480, PP0004499-PP0004607, PPSDK0000014, PPSDK0000930, PP0028495-PP0028498, PP0028499-PP0028500, PP0028501-PP0028507, PP0028508, PP0028509-PP0028513, PP0028514-PP0028518, PP0028519-PP0028521, PP0028522-PP0028523, PP0028524- PP0028525, PP0028526-PP0028530, PP0028531-PP0028532, PP0028533-PP0028536, PP0028537-PP0028538, PP0028539-PP0028543, PP0028544-PP0028557, PP0028558- PP0028561, PP0028562-PP0028567, PP0028568-PP0028573, PP0028574-PP0028579, PP0028580-PP0028583, PP0028584-PP0028595, PP0028596-PP0028597, PP0028598- PP0028599, PP0028600-PP0028611, PP0028612-PP0028614, PP0028615-

PP0028617, PP0028618-PP0028621, PP0028622-PP0028626, PP0028627-PP0028641, PP0028642- PP0028644, PP0028645-PP0028652, PP0028653-PP0028656, PP0028657-PP0028660, PP0028661-PP0028662, PP0028663-PP0028686, PP0028687-PP0028689, PP0028690- PP0028695, PP0028696-PP0028725, and PP0028726-PP0028730.

SECOND SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 9:

PayPal objects to this Interrogatory on the grounds that it is vague, ambiguous, and overly broad, particularly with respect to its usage of the phrases "for, on behalf of, at the direction of, or with the assistance of PayPal," "otherwise provided to PayPal," "for the benefit of," and "the purpose and operation of such code." PayPal objects to the phrase "Accused mPOS Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions."

IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

PayPal further objects to this Interrogatory as seeking publicly available information that IOENGINE could obtain as easily as could PayPal.

PayPal further objects to this Interrogatory as seeking information that is beyond the scope permitted under Federal Rules of Civil Procedure 26(b) and 34 because, as set forth in PayPal's motion to dismiss the complaint, IOENGINE has not properly pled any claim that puts the development of third-party applications at issue, and thus it is not entitled to discovery on this issue.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004400-PP0004480, PP0004499-PP0004607, PPSDK0000014, PPSDK0000930, PP0028495-PP0028498, PP0028499-PP0028500, PP0028501-PP0028507, PP0028508, PP0028509-PP0028513, PP0028514-PP0028518, PP0028519-PP0028521, PP0028522-PP0028523, PP0028524- PP0028525, PP0028526-PP0028530, PP0028531-PP0028532, PP0028533-PP0028536, PP0028537-PP0028538, PP0028539-PP0028543, PP0028544-PP0028557, PP0028558- PP0028561, PP0028562-PP0028567, PP0028568-PP0028573, PP0028574-PP0028579, PP0028580-PP0028583, PP0028584-PP0028595, PP0028596-PP0028597, PP0028598- PP0028599, PP0028600-PP0028611, PP0028612-PP0028614, PP0028615-PP0028617, PP0028618-PP0028621, PP0028622-PP0028626, PP0028627-PP0028641, PP0028642- PP0028644, PP0028645-PP0028652, PP0028653-PP0028656, PP0028657-PP0028660, PP0028661-PP0028662, PP0028663-PP0028686, PP0028687-PP0028689, PP0028690- PP0028695, PP0028696-PP0028725, PP0028726-PP0028730, PP0032588-PP0032620, PP0032951-PP0033528, PP0034147-PP0034196, and PP0033529-PP0034106.

THIRD SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 9:

PayPal objects to this Interrogatory on the grounds that it is vague, ambiguous, and overly broad, particularly with respect to its usage of the phrases "for, on behalf of, at the direction of, or with the assistance of PayPal," "otherwise provided to PayPal," "for the benefit of," and "the purpose and operation of such code."

PayPal further objects to the phrase "Accused mPOS Product" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they purport to relate to products or services other than those specifically identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." PayPal will interpret these terms to mean only the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

PayPal further objects to this Interrogatory as seeking publicly available information that IOENGINE could obtain as easily as could PayPal.

PayPal further objects to this Interrogatory as seeking information that is beyond the scope permitted under Federal Rules of Civil Procedure 26(b) and 34 because, as set forth in PayPal's motion to dismiss the complaint, IOENGINE has not properly pled any claim that puts the development of third-party applications at issue, and thus it is not entitled to discovery on this issue.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004400-PP0004480, PP0004499-PP0004607, PPSDK0000014, PPSDK0000930, PP0028495-PP0028498, PP0028499-PP0028500, PP0028501-PP0028507, PP0028508, PP0028509-PP0028513, PP0028514-PP0028518, PP0028519-PP0028521, PP0028522-PP0028523, PP0028524- PP0028525, PP0028526-PP0028530, PP0028531-PP0028532, PP0028533-PP0028536, PP0028537-PP0028538, PP0028539-PP0028543, PP0028544-PP0028557, PP0028558- PP0028561, PP0028562-PP0028567, PP0028568-PP0028573, PP0028574-PP0028579, PP0028580-PP0028583, PP0028584-PP0028595, PP0028596-PP0028597, PP0028598- PP0028599, PP0028600-PP0028611, PP0028612-PP0028614, PP0028615-

PP0028617, PP0028618-PP0028621, PP0028622-PP0028626, PP0028627-PP0028641, PP0028642- PP0028644, PP0028645-PP0028652, PP0028653-PP0028656, PP0028657-PP0028660, PP0028661-PP0028662, PP0028663-PP0028686, PP0028687-PP0028689, PP0028690- PP0028695, PP0028696-PP0028725, PP0028726-PP0028730, PP0032588-PP0032620, PP0032951-PP0033528, PP0034147-PP0034196, PP0033529-PP0034106, PP0194968-PP0195019, PP0194952-PP0194961, PP0195056-PP0195058, PP0195064-PP0195087, PP0195060-PP0195063, PP0195097-PP0195100, PP0195088-PP0195096, and PP0195131-PP0195158.

The accused card readers that are supplied to PayPal come pre-loaded with binary code that allow them to respond to commands from external sources. PayPal has provided physical samples to IOENGINE which include the binary code. *See* PPPS001- PPPS007. After a reasonable, good-faith investigation, PayPal did not identify any human readable source code in its possession, custody, or control provided by its card reader suppliers Miura, Ingenico, Verifone, or Datecs.

Various third-party POS partners have used PayPal SDKs and APIs to create their own software applications. Code developed by the POS Partners is designed to interoperate with PayPal's SDK to provide payment services to end users. After a reasonable, good-faith investigation, PayPal did not identify human readable source code for the POS Partner applications in PayPal's possession, custody, or control.

Dated: September 17, 2021

Of Counsel:

Jared Bobrow
jbobrow@orrick.com
Travis Jensen
tjensen@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
1000 Marsh Road
Menlo Park, CA 94025-1015
Telephone: +1 650 614 7400
Facsimile: +1 650 614 7401

Respectfully submitted,

MORRIS, NICHOLS, ARSHT &
TUNNELL LLP

/s/ Brian P. Egan
Jack B. Blumenfeld
jblumenfeld@mnat.com
Brian P. Egan
began@mnat.com
1201 North Market Street, Suite 1800
MORRIS, NICHOLS, ARSHT &
TUNNELL LLP
Wilmington, DE 19801
(302) 658-9200
jblumenfeld@mnat.com

CERTIFICATE OF SERVICE

I am employed in the County of San Mateo, State of California. I am over the age of 18 years old and not a party to this action. My business address is Orrick, Herrington & Sutcliffe LLP, 1000 Marsh Road, Menlo Park, California 94025. On September 17, 2021, I served the following document(s) entitled:

1. **PAYPAL'S SUPPLEMENTAL RESPONSES AND OBJECTIONS TO IOENGINE'S INTERROGATORY NOS. 1-4, 6-7, 9, 11-13, 15-19, AND 22-23**

on all interested parties to this action in the manner described as follows:

☒ **(VIA EMAIL)** On September 17, 2021, via electronic mail in Adobe PDF format the document(s) listed above to the electronic address(es) set forth below.

Noah M. Leibowitz - noah.leibowitz@dechert.com
Gregory T. Chuebon - greg.chuebon@dechert.com
Neal C. Belgam (No. 2721) - nbelgam@skjlaw.com
Eve H. Ormerod (No. 5369) - eormerod@skjlaw.com

Counsel for Plaintiff,
IOENGINE, LLC

I declare under penalty of perjury under the laws of the State of California and the United States that the foregoing is true and correct.

Executed on September 17, 2021, at Morgan Hill, California.

/s/ Katri Lahtinen
Katri Lahtinen

EXHIBIT 8

FILED UNDER SEAL

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IOENGINE, LLC,	)	
	)	
Plaintiff,	)	
	)	C.A. No. 18-452 (WCB)
v.	)	
	)	**CONFIDENTIAL – OUTSIDE**
PAYPAL HOLDINGS, INC.,	)	**ATTORNEYS' EYES ONLY –**
	)	**SOURCE CODE**
Defendant.	)	

PAYPAL'S SUPPLEMENTAL RESPONSES AND OBJECTIONS TO IOENGINE'S INTERROGATORY NOS. 1-5, 11-14, 17, 19, 22, AND 23

Pursuant to Federal Rules of Civil Procedure 26 and 33, Defendant PayPal Holdings, Inc. ("Defendant" or "PayPal") hereby objects and responds to Plaintiff IOENGINE, LLC's ("Plaintiff's" or "IOENGINE's") Interrogatory Nos. 1-5, 11-14, 17, 19, 22, and 23.

PRELIMINARY STATEMENT

PayPal has made a reasonable investigation for information responsive to IOENGINE's interrogatories based upon PayPal's current knowledge, information, and belief. PayPal's investigation is ongoing. PayPal's responses are made without prejudice to its right to revise, correct, supplement, or clarify its responses at any time pursuant to Federal Rule of Civil Procedure 26(e). PayPal reserves the right to make any use of, or to introduce at any hearing or trial, information responsive to these interrogatories that was discovered after the date of this response.

By providing responses, PayPal does not concede the relevancy of the subject matter of any interrogatory. PayPal reserves all objections or other questions as to the competency, relevance, materiality, privilege, or admissibility in any proceeding or trial for any purpose whatsoever of its responses provided herein. PayPal provides these written responses subject to the general and specific objections stated below, and subject to PayPal's right to object at any proceeding involving or relating to the subject matter of the requests responded to herein.

By providing responses, PayPal does not waive any of its rights and limitations on discovery set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, any other applicable law, rule, or order, and all other existing or future agreements of the parties concerning these proceedings, including discovery.

PayPal's responses contained herein and the information produced are provided solely for the purpose of discovery in this action. PayPal's responses are not to be used for any purpose outside the context of this lawsuit.

GENERAL OBJECTIONS

PayPal's responses to these interrogatories are subject to the following General Objections, which are hereby incorporated into PayPal's responses to the requests provided further below.

1. PayPal objects to these interrogatories to the extent they seek disclosure of information protected from disclosure under the attorney-client privilege, the doctrine of work product immunity, the common interest and joint defense privileges, or any other claim of privilege, law or rule ("Privileged Information"). PayPal will not disclose Privileged Information in response to these interrogatories, and PayPal's undertaking to respond should be understood to exclude Privileged Information. Any disclosure of Privileged Information is inadvertent and should not be construed to constitute a waiver. If PayPal withholds Privileged Information, it will log that information and respond in accordance with the terms of a suitable agreement to be negotiated pursuant to Delaware Default Standard for Discovery Paragraph 1.d.

2. PayPal objects to these interrogatories to the extent they seek disclosure of information protected by the rights of privacy of third parties or to the extent they seek disclosure of confidential information of third parties to which PayPal owes an obligation of confidentiality

or duty to prevent disclosure, including but not limited to information subject to confidentiality agreements, non-disclosure agreements, protective orders, or the General Data Protection Regulation in the European Union.

3. PayPal objects to these interrogatories to the extent they seek to impose obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. For example, PayPal objects to these requests to the extent they are overly broad, unduly burdensome, harassing, and/or oppressive and to the extent they seek information or documents neither relevant to a claim or defense of any party nor proportional to the needs of the case, considering the importance of the issues at stake in the action, the amount in controversy, the parties' relative access to relevant information, the parties' resources, the importance of the discovery in resolving the issues, and whether the burden or expense of the proposed discovery outweighs its likely benefit.

4. PayPal objects to these interrogatories to the extent they seek information or documents not in its possession, custody, or control or purport to require PayPal to respond on behalf of another entity or call for information not within PayPal's own knowledge.

5. PayPal objects to these interrogatories to the extent they purport to shift the burden of proving any fact or issue borne by IOENGINE onto PayPal.

6. PayPal objects to these interrogatories to the extent they seek publicly available information, information already within IOENGINE's possession, custody, or control, or information that IOENGINE could obtain as easily as could PayPal.

7. PayPal objects to these interrogatories to the extent they attempt to impose a duty on PayPal to undertake a search for information beyond a reasonable search of PayPal's files where PayPal would expect to find information responsive to the requests.

GENERAL OBJECTIONS TO "DEFINITIONS" AND "INSTRUCTIONS"

1. In addition to the General Objections above, PayPal's responses to these interrogatories are subject to the following objections to the "Definitions" and "Instructions" provided with the requests, each of which are hereby incorporated by reference into PayPal's responses below. If PayPal objects to a definition of a term and that term is used in the definition of a subsequent term, PayPal's objections to the term used in the subsequent definition are incorporated by reference therein.

2. PayPal objects to IOENGINE's "Definitions" and "Instructions" to the extent that they contradict and/or impose obligations upon PayPal in excess of or different from the duties and obligations imposed by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

3. PayPal objects to the definition of the terms "PayPal," "Zettle," "You," and "Your" on the ground that they are overly broad and unduly burdensome, seek information that is privileged or immune from discovery, and include persons or entities that PayPal has no control over and, therefore, PayPal does not have possession, custody, or control of information of any such persons or entities. PayPal will interpret each of these terms—"PayPal," "Zettle," "You," and "Your," and similar terms—to mean PayPal Holdings, Inc.

4. PayPal objects to the definition of the terms "Document" and "All Documents" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definitions of the terms "Document" and "All Documents" to the extent they call for the production of email and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

5. PayPal objects to the definition of the term "Communication" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definition of the term "Communication" to the extent it calls for the production of e-mail, Internet communications, and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

6. PayPal objects to the definitions of the terms "Thing," "Relate," "Relate To," "Relating To," "Concern," "Concerning," and "Person" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order.

7. PayPal objects to the definition of the terms "Identify" and "Describe" as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent identification or description is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

8. PayPal objects to the definitions of the terms "Accused mPOS Product," "Accused PayPal Product(s)," "Ingenico-Supplied PayPal Products," "Accused PayPal Devices," "PayPal Here App," "PayPal POS Partner App," "Accused Mobile Payment Apps," and "PayPal Here Service," as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they purport to relate to products or services other than those specifically identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret

these terms to mean only the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

9. PayPal objects to the definitions of the terms "Zettle Service" and "Zettle App" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret these terms to mean only the Zettle App (*see* https://play.google.com/store/apps/details?id=com.izettle.android&hl=en_US&gl=US and https://apps.apple.com/us/app/paypal-zettle-point-of-sale/id447785763) and Zettle Reader as released in the United States (*see* www.paypal.com/us/business/pos/payments/card-reader).

10. PayPal objects to the definition of the term "Transaction Revenue" as overly broad, unduly burdensome, not proportional to the needs of the case, vague, and ambiguous to the extent it purports to include fees paid "for the benefit of PayPal" and all fees "Relating To any Accused PayPal Products." PayPal further objects to IOENGINE's request for transaction revenue as not relevant to any party's claim or defense and not proportional to the needs of the case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention.

11. PayPal objects to the definition of the term "Accused Functionality" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it purports to relate to functionalities other than those that are identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret this term to mean those functionalities.

12. PayPal objects to the definition of the terms "POS Partner App," "POS Partner," and "POS Partners" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret the term "POS Partner App" to mean only the Lavu Point of Sale App,

TouchBistro App, Vend Register App, ERPLY cloud-based POS software, Brightpearl POS system, and TouchPoint POS system after June 2015. PayPal will interpret the terms "POS Partner" and "POS Partners" to mean only the providers of the aforementioned POS Partner Apps.

13. PayPal objects to the definition of the term "Mobile Credit Card Processing Market" as vague, overly broad, unduly burdensome, and not proportional to the needs of the case, as it merely points to ten pages of a hearing transcript that did not "ascribe" any specific meaning to the term, as IOENGINE incorrectly states. As the Court clearly stated to Ingenico's counsel with respect to this issue, "[a]ll you can be expected to do is report what you have and you are not expected to go out and produce information you don't have." 2021-03-31 Hearing Tr. at 75:6-8.

14. PayPal objects to the definition of the term "Compatible Terminal" to the extent that it calls for a legal conclusion and/or improperly suggests that PayPal technology is used in conjunction with any device that infringes any claim of any Patent-in-Suit. PayPal specifically denies any such alleged infringement. PayPal further objects to the definition of this term to be extent it suggests that the rights afforded by the Asserted Patents extend beyond the subject matter defined in the claims. PayPal will not provide responses that relate to the structure or function of any device that is not provided by PayPal, except for the operation of the "PayPal Here App" and any properly accused "PayPal POS Partner Apps," as those terms were objected to above.

15. PayPal objects to "Instructions" Paragraphs 3-11 to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

16. PayPal further objects to objects to the identifying information and descriptions requested in "Instructions" Paragraphs 3-11 as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent any of the identifying information or

descriptions requested is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

17. PayPal objects to "Instructions" Paragraph 12 as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it suggests that all information relating to the time period from six years before the filing of this Action to the present is relevant, no matter what issues it relates to. PayPal will respond with respect to the time period that is relevant to each request.

GENERAL FINANCIAL INFORMATION OBJECTION

1. IOENGINE's requests and interrogatories regarding financial information are overbroad, unduly burdensome, and not proportional to the needs of the case to the extent they seek financial information other than:

- information sufficient to show quarterly revenue and costs of goods sold for U.S. sales of the products identified in PayPal's response to Interrogatory No. 1 since June 2015;
- information sufficient to show the number of units sold on a quarterly basis in the U.S. for the products identified in PayPal's response to Interrogatory No. 1 since June 2015; and
- PayPal's annual Form 10-K filings since 2015.

2. PayPal objects to IOENGINE's requests and interrogatories requesting revenue information related to providing any services, including transaction revenue, as not relevant to any party's claim or defense and not proportional to the needs of the case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention. PayPal objects to any requests or interrogatories relating to PayPal's financial information as overbroad, unduly burdensome, and not proportional

to the needs of the case, to the extent they require PayPal to provide information or produce material beyond that in the bullet-point list, *supra*.

SPECIFIC OBJECTIONS AND RESPONSES

INTERROGATORY NO. 1:

Identify by name, version number, and/or any other designation, each version or type of any Accused PayPal Product made (in whole or in part), used, sold, offered for sale, imported into, or otherwise distributed within the United States, and separately for each such version, state the quantities made, used, sold, offered for sale, imported into, or otherwise distributed within the United States by month from January 1, 2012 to present.

RESPONSE TO INTERROGATORY NO. 1:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States." PayPal further objects to the term "Accused PayPal Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, and PayPal Chip and Tap Reader. PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until September 2013. PayPal incorporates by reference its General Financial Information Objection. PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the products named in IOENGINE's Complaint or Delaware Default Standard for Discovery Paragraph 4.a. disclosures.

ii. The "Model Numbers" column lists model number(s) of the products in the "Name" column.

iii. The "Other Designation" column lists additional name(s) of the products in the "Name" column.

Name	**Model Numbers**	**Other Designation(s)**
PayPal Chip Card Reader	M010-PROD10-v2-7	M010, M10, Chip and Pin, C&P
PayPal Mobile Card Reader	G3X, G4XD, G5X	Swiper, Triangle Reader, Magnetic Stripe Reader, PPH 1.0, PPH2.0, PPH2.2
PayPal Chip and Swipe Reader	Moby/3000, M0B30AA	Chip and Swipe, C&S, Moby, Moby 3000
PayPal Chip and Tap Reader	RP457c-BT, RP457c-0BT8901B	Chip and Tap, C&T, RP457, P457c

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from documents that PayPal will produce.

FIRST SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 1:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States." PayPal further objects to the term "Accused PayPal Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, and PayPal Chip and Tap Reader. PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until September 2013. PayPal incorporates by reference its General Financial Information Objection. PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the products named in IOENGINE's Complaint or Delaware Default Standard for Discovery Paragraph 4.a. disclosures.

ii. The "Model Numbers" column lists model number(s) of the products in the "Name" column.

iii. The "Other Designation" column lists additional name(s) of the products in the "Name" column.

Name	**Model Numbers**	**Other Designation(s)**
PayPal Chip Card Reader	M010-PROD10-v2-7	M010, M10, Chip and Pin, C&P
PayPal Mobile Card Reader	G3X, G4XD, G5X	Swiper, Triangle Reader, Magnetic Stripe Reader, PPH 1.0, PPH2.0, PPH2.2
PayPal Chip and Swipe Reader	Moby/3000, M0B30AA	Chip and Swipe, C&S, Moby, Moby 3000

Name	Model Numbers	Other Designation(s)
PayPal Chip and Tap Reader	RP457c-BT, RP457c-0BT8901B	Chip and Tap, C&T, RP457, P457c

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004497-PP0004498, and PP0006186.

SECOND SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 1:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States." PayPal further objects to the term "Accused PayPal Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and the Verifone Reader. PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015. PayPal incorporates by reference its General Financial Information Objection. PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the products named in IOENGINE's Complaint or Delaware Default Standard for Discovery Paragraph 4.a. disclosures.

ii. The "Model Numbers" column lists model number(s) of the products in the "Name" column.

iii. The "Other Designation" column lists additional name(s) of the products in the "Name" column.

Name	**Model Numbers**	**Other Designation(s)**
PayPal Chip Card Reader	M010-PROD10-v2-7	M010, M10, Chip and Pin, C&P
PayPal Mobile Card Reader	G3X, G4XD, G5X	Swiper, Triangle Reader, Magnetic Stripe Reader, PPH 1.0, PPH2.0, PPH2.2
PayPal Chip and Swipe Reader	Moby/3000, M0B30AA	Chip and Swipe, C&S, Moby, Moby 3000
PayPal Chip and Tap Reader	RP457c-BT, RP457c-0BT8901B	Chip and Tap, C&T, RP457, P457c
VeriFone Reader	P400 Pro	P400+

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004497-PP0004498, PP0006186, and PP0031119.

THIRD SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 1:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States." PayPal further objects to the term "Accused PayPal Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and the Verifone Reader. PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015. PayPal incorporates by reference its General Financial Information Objection. PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the products named in IOENGINE's Complaint or Delaware Default Standard for Discovery Paragraph 4.a. disclosures.

ii. The "Model Numbers" column lists model number(s) of the products in the "Name" column.

iii. The "Other Designation" column lists additional name(s) of the products in the "Name" column.

Name	Model Numbers	Other Designation(s)
PayPal Chip Card Reader	M010-PROD10-v2-7	M010, M10, Chip and Pin, C&P
PayPal Mobile Card Reader	G3X, G4XD, G5X	Swiper, Triangle Reader, Magnetic Stripe Reader, PPH 1.0, PPH2.0, PPH2.2
PayPal Chip and Swipe Reader	Moby/3000, M0B30AA	Chip and Swipe, C&S, Moby, Moby 3000
PayPal Chip and Tap Reader	RP457c-BT, RP457c-0BT8901B	Chip and Tap, C&T, RP457, P457c
VeriFone Reader	P400 Pro	P400+

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0004497-PP0004498, PP0006186, PP0031119, PP0200471, PP0200284, PP0026740; PP0040853, PP0044865, PP0044875, PP0046601, PP0048078, PP0120864, PP0145952, and PP0196213.

FOURTH SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 1:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States." PayPal further objects to the term "Accused PayPal Product" and specifically incorporates paragraph 8 of its General Objections to "Definitions" and "Instructions." IOENGINE's Complaint and Delaware Default Standard for Discovery Paragraph 4.a. disclosures do not adequately identify the accused PayPal products, or the functionalities and/or structures therein that are accused of infringement. PayPal will respond only as to the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and the Verifone Reader. PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015. PayPal incorporates by reference its General Financial Information Objection. PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the products named in IOENGINE's Complaint or Delaware Default Standard for Discovery Paragraph 4.a. disclosures.

ii. The "Model Numbers" column lists model number(s) of the products in the "Name" column.

iii. The "Other Designation" column lists additional name(s) of the products in the "Name" column.

Name	Model Numbers	Other Designation(s)
PayPal Chip Card Reader	M010-PROD10-v2-7	M010, M10, Chip and Pin, C&P
PayPal Mobile Card Reader	G3X, G4XD, G5X	Swiper, Triangle Reader, Magnetic Stripe Reader, PPH 1.0, PPH2.0, PPH2.2
PayPal Chip and Swipe Reader	Moby/3000, M0B30AA	Chip and Swipe, C&S, Moby, Moby 3000
PayPal Chip and Tap Reader	RP457c-BT, RP457c-0BT8901B	Chip and Tap, C&T, RP457, P457c
VeriFone Reader	P400 Pro	P400+

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents:

PP0004497-PP0004498, PP0006186, PP0031119, PP0200471, PP0200284, PP0026740; PP0040853, PP0044865, PP0044875, PP0046601, PP0048078, PP0120864, PP0145952, PP0196213, PP1000567, and PP1000569.

[redacted]

[redacted]

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Brian P. Egan

Jack B. Blumenfeld (#1014)
Brian P. Egan (#6227)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
(302) 658-9200
jblumenfeld@morrisnichols.com
began@morrisnichols.com

Attorneys for Defendant

OF COUNSEL:

Jared Bobrow
Travis Jensen
ORRICK, HERRINGTON & SUTCLIFFE LLP
1000 Marsh Road
Menlo Park, CA 94025-1015
(650) 614-7400

October 20, 2021

CERTIFICATE OF SERVICE

I hereby certify that on October 20, 2021, copies of the foregoing were caused to be served upon the following in the manner indicated:

Neal C. Belgam, Esquire
Eve H. Ormerod, Esquire
SMITH, KATZENSTEIN & JENKINS, LLP
1000 West Street, Suite 1501
Wilmington, DE 19801
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Noah M. Leibowitz, Esquire
Gregory T. Chuebon, Esquire
DECHERT LLP
Three Bryant Park
1095 Avenue of the Americas
New York, NY 10036-6797
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Derek J. Brader, Esquire
Luke M. Reilly, Esquire
Michael A. Fisher, Esquire
Judah Bellin, Esquire
DECHERT LLP
Cira Centre
2929 Arch Street
Philadelphia, PA 19104-2808
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Michael H. Joshi, Esquire
DECHERT LLP
3000 El Camino Real
Five Palo Alto Square, Suite 650
Palo Alto, CA 94306
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

/s/ Brian P. Egan

Brian P. Egan (#6227)

EXHIBIT 9

FILED UNDER SEAL

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IOENGINE, LLC,	)	
	)	
Plaintiff,	)	
	)	C.A. No. 18-452 (WCB)
v.	)	
	)	**CONFIDENTIAL – OUTSIDE**
PAYPAL HOLDINGS, INC.,	)	**ATTORNEYS' EYES ONLY –**
	)	**SOURCE CODE**
Defendant.	)	

PAYPAL'S SUPPLEMENTAL RESPONSES AND OBJECTIONS TO IOENGINE'S INTERROGATORY NOS. 1-5, 11-14, 17, 19, 22, AND 23

Pursuant to Federal Rules of Civil Procedure 26 and 33, Defendant PayPal Holdings, Inc. ("Defendant" or "PayPal") hereby objects and responds to Plaintiff IOENGINE, LLC's ("Plaintiff's" or "IOENGINE's") Interrogatory Nos. 1-5, 11-14, 17, 19, 22, and 23.

PRELIMINARY STATEMENT

PayPal has made a reasonable investigation for information responsive to IOENGINE's interrogatories based upon PayPal's current knowledge, information, and belief. PayPal's investigation is ongoing. PayPal's responses are made without prejudice to its right to revise, correct, supplement, or clarify its responses at any time pursuant to Federal Rule of Civil Procedure 26(e). PayPal reserves the right to make any use of, or to introduce at any hearing or trial, information responsive to these interrogatories that was discovered after the date of this response.

By providing responses, PayPal does not concede the relevancy of the subject matter of any interrogatory. PayPal reserves all objections or other questions as to the competency, relevance, materiality, privilege, or admissibility in any proceeding or trial for any purpose whatsoever of its responses provided herein. PayPal provides these written responses subject to the general and specific objections stated below, and subject to PayPal's right to object at any proceeding involving or relating to the subject matter of the requests responded to herein.

By providing responses, PayPal does not waive any of its rights and limitations on discovery set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, any other applicable law, rule, or order, and all other existing or future agreements of the parties concerning these proceedings, including discovery.

PayPal's responses contained herein and the information produced are provided solely for the purpose of discovery in this action. PayPal's responses are not to be used for any purpose outside the context of this lawsuit.

GENERAL OBJECTIONS

PayPal's responses to these interrogatories are subject to the following General Objections, which are hereby incorporated into PayPal's responses to the requests provided further below.

1. PayPal objects to these interrogatories to the extent they seek disclosure of information protected from disclosure under the attorney-client privilege, the doctrine of work product immunity, the common interest and joint defense privileges, or any other claim of privilege, law or rule ("Privileged Information"). PayPal will not disclose Privileged Information in response to these interrogatories, and PayPal's undertaking to respond should be understood to exclude Privileged Information. Any disclosure of Privileged Information is inadvertent and should not be construed to constitute a waiver. If PayPal withholds Privileged Information, it will log that information and respond in accordance with the terms of a suitable agreement to be negotiated pursuant to Delaware Default Standard for Discovery Paragraph 1.d.

2. PayPal objects to these interrogatories to the extent they seek disclosure of information protected by the rights of privacy of third parties or to the extent they seek disclosure of confidential information of third parties to which PayPal owes an obligation of confidentiality

or duty to prevent disclosure, including but not limited to information subject to confidentiality agreements, non-disclosure agreements, protective orders, or the General Data Protection Regulation in the European Union.

3. PayPal objects to these interrogatories to the extent they seek to impose obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. For example, PayPal objects to these requests to the extent they are overly broad, unduly burdensome, harassing, and/or oppressive and to the extent they seek information or documents neither relevant to a claim or defense of any party nor proportional to the needs of the case, considering the importance of the issues at stake in the action, the amount in controversy, the parties' relative access to relevant information, the parties' resources, the importance of the discovery in resolving the issues, and whether the burden or expense of the proposed discovery outweighs its likely benefit.

4. PayPal objects to these interrogatories to the extent they seek information or documents not in its possession, custody, or control or purport to require PayPal to respond on behalf of another entity or call for information not within PayPal's own knowledge.

5. PayPal objects to these interrogatories to the extent they purport to shift the burden of proving any fact or issue borne by IOENGINE onto PayPal.

6. PayPal objects to these interrogatories to the extent they seek publicly available information, information already within IOENGINE's possession, custody, or control, or information that IOENGINE could obtain as easily as could PayPal.

7. PayPal objects to these interrogatories to the extent they attempt to impose a duty on PayPal to undertake a search for information beyond a reasonable search of PayPal's files where PayPal would expect to find information responsive to the requests.

GENERAL OBJECTIONS TO "DEFINITIONS" AND "INSTRUCTIONS"

1. In addition to the General Objections above, PayPal's responses to these interrogatories are subject to the following objections to the "Definitions" and "Instructions" provided with the requests, each of which are hereby incorporated by reference into PayPal's responses below. If PayPal objects to a definition of a term and that term is used in the definition of a subsequent term, PayPal's objections to the term used in the subsequent definition are incorporated by reference therein.

2. PayPal objects to IOENGINE's "Definitions" and "Instructions" to the extent that they contradict and/or impose obligations upon PayPal in excess of or different from the duties and obligations imposed by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

3. PayPal objects to the definition of the terms "PayPal," "Zettle," "You," and "Your" on the ground that they are overly broad and unduly burdensome, seek information that is privileged or immune from discovery, and include persons or entities that PayPal has no control over and, therefore, PayPal does not have possession, custody, or control of information of any such persons or entities. PayPal will interpret each of these terms—"PayPal," "Zettle," "You," and "Your," and similar terms—to mean PayPal Holdings, Inc.

4. PayPal objects to the definition of the terms "Document" and "All Documents" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definitions of the terms "Document" and "All Documents" to the extent they call for the production of email and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

5. PayPal objects to the definition of the term "Communication" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that it seeks to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order. Further, PayPal objects to the definition of the term "Communication" to the extent it calls for the production of e-mail, Internet communications, and any other electronic communications as overly broad, unduly burdensome, and not proportional to the needs of the case.

6. PayPal objects to the definitions of the terms "Thing," "Relate," "Relate To," "Relating To," "Concern," "Concerning," and "Person" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, or any other applicable law, rule, or order.

7. PayPal objects to the definition of the terms "Identify" and "Describe" as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent identification or description is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

8. PayPal objects to the definitions of the terms "Accused mPOS Product," "Accused PayPal Product(s)," "Ingenico-Supplied PayPal Products," "Accused PayPal Devices," "PayPal Here App," "PayPal POS Partner App," "Accused Mobile Payment Apps," and "PayPal Here Service," as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent that they purport to relate to products or services other than those specifically identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret

these terms to mean only the PayPal Chip Card Reader, PayPal Mobile Card Reader, PayPal Chip and Swipe Reader, PayPal Chip and Tap Reader, and PayPal Here App.

9. PayPal objects to the definitions of the terms "Zettle Service" and "Zettle App" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret these terms to mean only the Zettle App (*see* https://play.google.com/store/apps/details?id=com.izettle.android&hl=en_US&gl=US and https://apps.apple.com/us/app/paypal-zettle-point-of-sale/id447785763) and Zettle Reader as released in the United States (*see* www.paypal.com/us/business/pos/payments/card-reader).

10. PayPal objects to the definition of the term "Transaction Revenue" as overly broad, unduly burdensome, not proportional to the needs of the case, vague, and ambiguous to the extent it purports to include fees paid "for the benefit of PayPal" and all fees "Relating To any Accused PayPal Products." PayPal further objects to IOENGINE's request for transaction revenue as not relevant to any party's claim or defense and not proportional to the needs of the case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention.

11. PayPal objects to the definition of the term "Accused Functionality" as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it purports to relate to functionalities other than those that are identified in IOENGINE's November 4, 2020 Supplemental Infringement Contentions. PayPal will interpret this term to mean those functionalities.

12. PayPal objects to the definition of the terms "POS Partner App," "POS Partner," and "POS Partners" as overly broad, unduly burdensome, and not proportional to the needs of the case. PayPal will interpret the term "POS Partner App" to mean only the Lavu Point of Sale App,

TouchBistro App, Vend Register App, ERPLY cloud-based POS software, Brightpearl POS system, and TouchPoint POS system after June 2015. PayPal will interpret the terms "POS Partner" and "POS Partners" to mean only the providers of the aforementioned POS Partner Apps.

13. PayPal objects to the definition of the term "Mobile Credit Card Processing Market" as vague, overly broad, unduly burdensome, and not proportional to the needs of the case, as it merely points to ten pages of a hearing transcript that did not "ascribe" any specific meaning to the term, as IOENGINE incorrectly states. As the Court clearly stated to Ingenico's counsel with respect to this issue, "[a]ll you can be expected to do is report what you have and you are not expected to go out and produce information you don't have." 2021-03-31 Hearing Tr. at 75:6-8.

14. PayPal objects to the definition of the term "Compatible Terminal" to the extent that it calls for a legal conclusion and/or improperly suggests that PayPal technology is used in conjunction with any device that infringes any claim of any Patent-in-Suit. PayPal specifically denies any such alleged infringement. PayPal further objects to the definition of this term to be extent it suggests that the rights afforded by the Asserted Patents extend beyond the subject matter defined in the claims. PayPal will not provide responses that relate to the structure or function of any device that is not provided by PayPal, except for the operation of the "PayPal Here App" and any properly accused "PayPal POS Partner Apps," as those terms were objected to above.

15. PayPal objects to "Instructions" Paragraphs 3-11 to the extent that they seek to impose on PayPal obligations in addition to or inconsistent with those set forth in the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

16. PayPal further objects to objects to the identifying information and descriptions requested in "Instructions" Paragraphs 3-11 as overly broad, unduly burdensome, and not proportional to the needs of the case. To the extent any of the identifying information or

descriptions requested is necessary, if it all, PayPal will provide the identifying information or description that is required by the Federal Rules of Civil Procedure, the Local Rules of this Court, and any other applicable rules or law.

17. PayPal objects to "Instructions" Paragraph 12 as overly broad, unduly burdensome, and not proportional to the needs of the case to the extent it suggests that all information relating to the time period from six years before the filing of this Action to the present is relevant, no matter what issues it relates to. PayPal will respond with respect to the time period that is relevant to each request.

GENERAL FINANCIAL INFORMATION OBJECTION

1. IOENGINE's requests and interrogatories regarding financial information are overbroad, unduly burdensome, and not proportional to the needs of the case to the extent they seek financial information other than:

- information sufficient to show quarterly revenue and costs of goods sold for U.S. sales of the products identified in PayPal's response to Interrogatory No. 1 since June 2015;
- information sufficient to show the number of units sold on a quarterly basis in the U.S. for the products identified in PayPal's response to Interrogatory No. 1 since June 2015; and
- PayPal's annual Form 10-K filings since 2015.

2. PayPal objects to IOENGINE's requests and interrogatories requesting revenue information related to providing any services, including transaction revenue, as not relevant to any party's claim or defense and not proportional to the needs of the case. For instance, the amount of revenue a transaction(s) involves has no relationship to whether infringement has or has not occurred or the value of the alleged invention. PayPal objects to any requests or interrogatories relating to PayPal's financial information as overbroad, unduly burdensome, and not proportional

to the needs of the case, to the extent they require PayPal to provide information or produce material beyond that in the bullet-point list, *supra*.

SPECIFIC OBJECTIONS AND RESPONSES

[redacted]

[redacted]

INTERROGATORY NO. 11:

Identify by name, version number, and/or any other designation each version or type of any Zettle Reader made (in whole or in part), used, sold, offered for sale, imported into, or otherwise distributed within the United States, and separately for each such version, state the quantities made, used, sold, offered for sale, imported into, or otherwise distributed within the United States by month from January 1, 2012 to present.

RESPONSE TO INTERROGATORY NO. 11:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States."

PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015.

PayPal incorporates by reference its General Financial Information Objection.

PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the consumer facing name of the Zettle Reader.

ii. The "Model Numbers" column lists model number(s) of the Zettle Reader

iii. The "Other Designation" column lists additional name(s) of the Zettle Reader in the "Name" column.

Name	Model Numbers	Other Designation(s)
PayPal Zettle Reader	Zettle Reader 2	Zettle Reader 2 iZettle Reader 2 Zettle Reader 2.0 iZettle Reader 2.0 V2 Customized BluePad 50 Datecs_2

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from documents that PayPal will produce.

FIRST SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 11:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States."

PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015.

PayPal incorporates by reference its General Financial Information Objection.

PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the consumer facing name of the Zettle Reader.

ii. The "Model Numbers" column lists model number(s) of the Zettle Reader

iii. The "Other Designation" column lists additional name(s) of the Zettle Reader in the "Name" column.

Name	**Model Numbers**	**Other Designation(s)**
PayPal Zettle Reader	Zettle Reader 2	Zettle Reader 2 iZettle Reader 2 Zettle Reader 2.0 iZettle Reader 2.0 V2 Customized BluePad 50 Datecs_2

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0200471, PP0200284, and PP0196213.

SECOND SUPPLEMENTAL RESPONSE TO INTERROGATORY NO. 11:

PayPal objects to this Interrogatory on the grounds that it is vague and overly broad, particularly with respect to its usage of the phrase "otherwise distributed within the United States."

PayPal further objects to this Interrogatory as seeking information relating to a time period that is not relevant to the claims or defenses of any party and not proportional to the needs of the case, because the earliest-issued asserted patent did not issue until June 2015.

PayPal incorporates by reference its General Financial Information Objection.

PayPal further objects to this Interrogatory as improperly characterized as a single interrogatory because it contains multiple subparts that constitute separate interrogatories, each of which count toward IOENGINE's interrogatory limit.

Subject to and without waiving these foregoing General and Specific objections, PayPal responds as follows:

i. The "Name" column lists the consumer facing name of the Zettle Reader.

ii. The "Model Numbers" column lists model number(s) of the Zettle Reader

iii. The "Other Designation" column lists additional name(s) of the Zettle Reader in the "Name" column.

Name	**Model Numbers**	**Other Designation(s)**
PayPal Zettle Reader	Zettle Reader 2	Zettle Reader 2 iZettle Reader 2 Zettle Reader 2.0 iZettle Reader 2.0 V2 Customized BluePad 50 Datecs_2

PayPal further responds that, in accordance with Fed. R. Civ. P. 33(d), information responsive to this Interrogatory may be derived or ascertained from the following documents: PP0200471, PP0200284, PP0196213, PP1000567, and PP1000569.

[redacted]

[redacted]

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Brian P. Egan

Jack B. Blumenfeld (#1014)
Brian P. Egan (#6227)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
(302) 658-9200
jblumenfeld@morrisnichols.com
began@morrisnichols.com

Attorneys for Defendant

OF COUNSEL:

Jared Bobrow
Travis Jensen
ORRICK, HERRINGTON & SUTCLIFFE LLP
1000 Marsh Road
Menlo Park, CA 94025-1015
(650) 614-7400

October 20, 2021

CERTIFICATE OF SERVICE

I hereby certify that on October 20, 2021, copies of the foregoing were caused to be served upon the following in the manner indicated:

Neal C. Belgam, Esquire
Eve H. Ormerod, Esquire
SMITH, KATZENSTEIN & JENKINS, LLP
1000 West Street, Suite 1501
Wilmington, DE 19801
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Noah M. Leibowitz, Esquire
Gregory T. Chuebon, Esquire
DECHERT LLP
Three Bryant Park
1095 Avenue of the Americas
New York, NY 10036-6797
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Derek J. Brader, Esquire
Luke M. Reilly, Esquire
Michael A. Fisher, Esquire
Judah Bellin, Esquire
DECHERT LLP
Cira Centre
2929 Arch Street
Philadelphia, PA 19104-2808
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

Michael H. Joshi, Esquire
DECHERT LLP
3000 El Camino Real
Five Palo Alto Square, Suite 650
Palo Alto, CA 94306
Attorneys for Plaintiff

VIA ELECTRONIC MAIL

/s/ Brian P. Egan

Brian P. Egan (#6227)

EXHIBIT 10

US 20040039932A1

(19) **United States**
(12) **Patent Application Publication** (10) **Pub. No.: US 2004/0039932 A1**
Elazar et al. (43) **Pub. Date: Feb. 26, 2004**

(54) **APPARATUS, SYSTEM AND METHOD FOR SECURING DIGITAL DOCUMENTS IN A DIGITAL APPLIANCE**

(76) Inventors: **Gidon Elazar**, Tsur-Igal (IL); **Dan Harkabi**, Lachish (IL); **Nehemiah Weingarten**, Ramat Gan (IL)

Correspondence Address:
WILSON SONSINI GOODRICH & ROSATI
650 PAGE MILL ROAD
PALO ALTO, CA 943041050

(21) Appl. No.: **10/227,155**

(22) Filed: **Aug. 23, 2002**

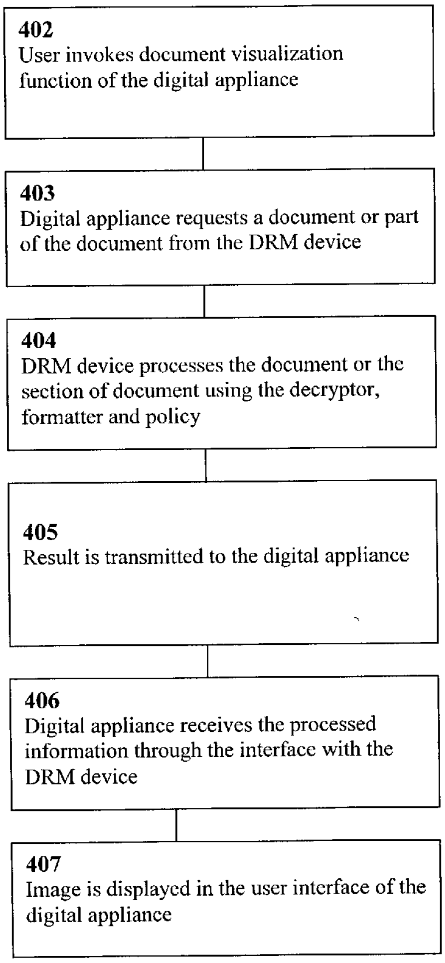
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/200**; 705/59

(57) **ABSTRACT**

Various embodiments include an apparatus and a method to secure protected digital document content from tampering by their user, such as unauthenticated use or use violating a policy of the digital document. The digital document file can be transferred from a network node such as a web site server to a digital appliance, such as a computer, in encrypted form. The digital document file can be resident already on a device, and/or be transferred into a device that is connected to the digital appliance. The device (hereafter a DRM device) can internally store the digital document or part of the document. The DRM device may decrypt the digital document when requested to do so. The device may further format the content for usage, for example, convert text into its graphic bitmap representation. Device formatting can include sending plain text data to the digital appliance. The device may further process degradation to the resulted file, for example, reduce the resolution of the graphic representation. The digital appliance uploads the result of the processing or sections of the result of the processing for user access via the digital appliance.


402
User invokes document visualization function of the digital appliance
403
Digital appliance requests a document or part of the document from the DRM device
404
DRM device processes the document or the section of document using the decryptor, formatter and policy
405
Result is transmitted to the digital appliance
406
Digital appliance receives the processed information through the interface with the DRM device
407
Image is displayed in the user interface of the digital appliance

Ingenico Inc. v. IOENGINE LLC
DX-363
C.A. No. 18-cv-00826-WCB

Elazar Exhibit
6

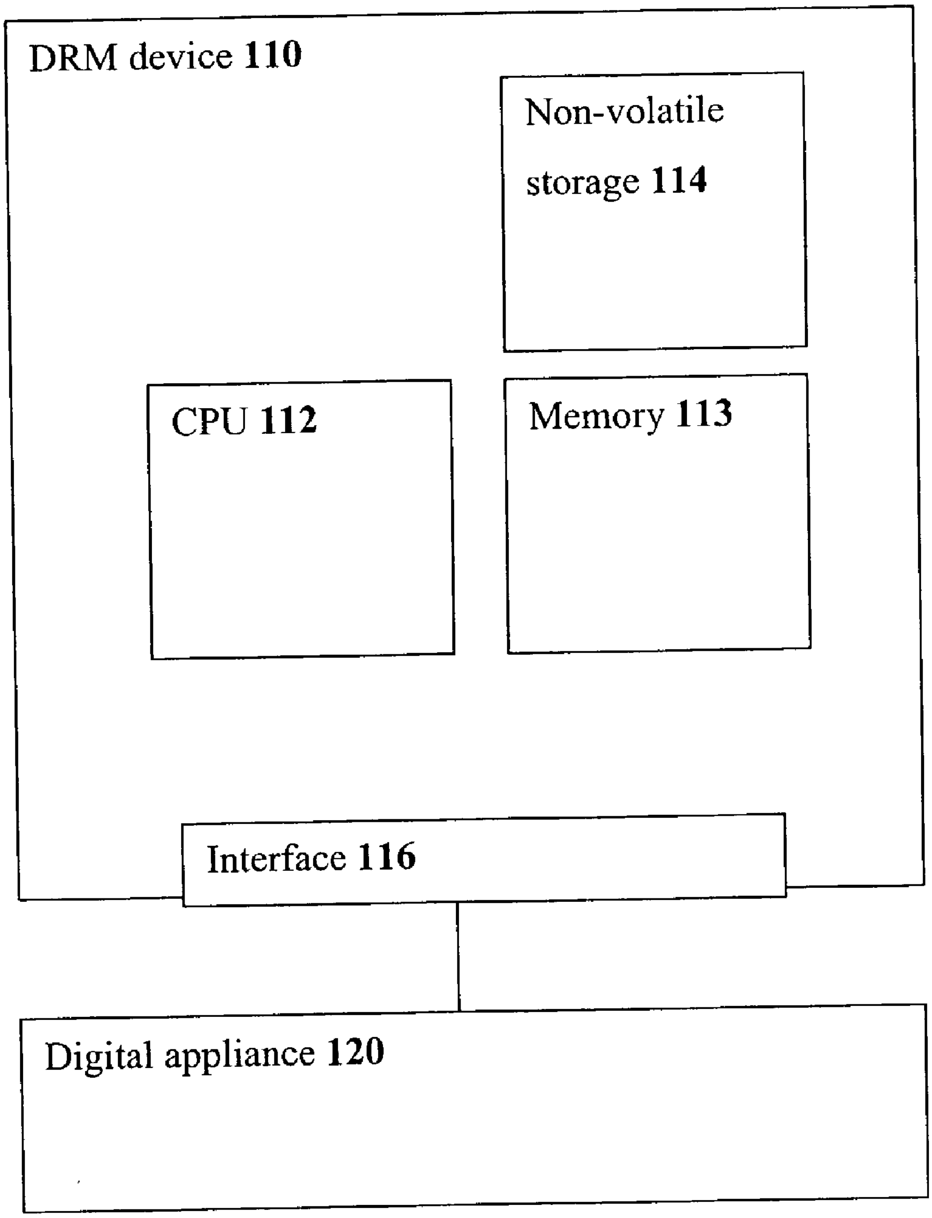

DRM device 110
Non-volatile storage 114
CPU 112
Memory 113
Interface 116
Digital appliance 120

FIG. 1

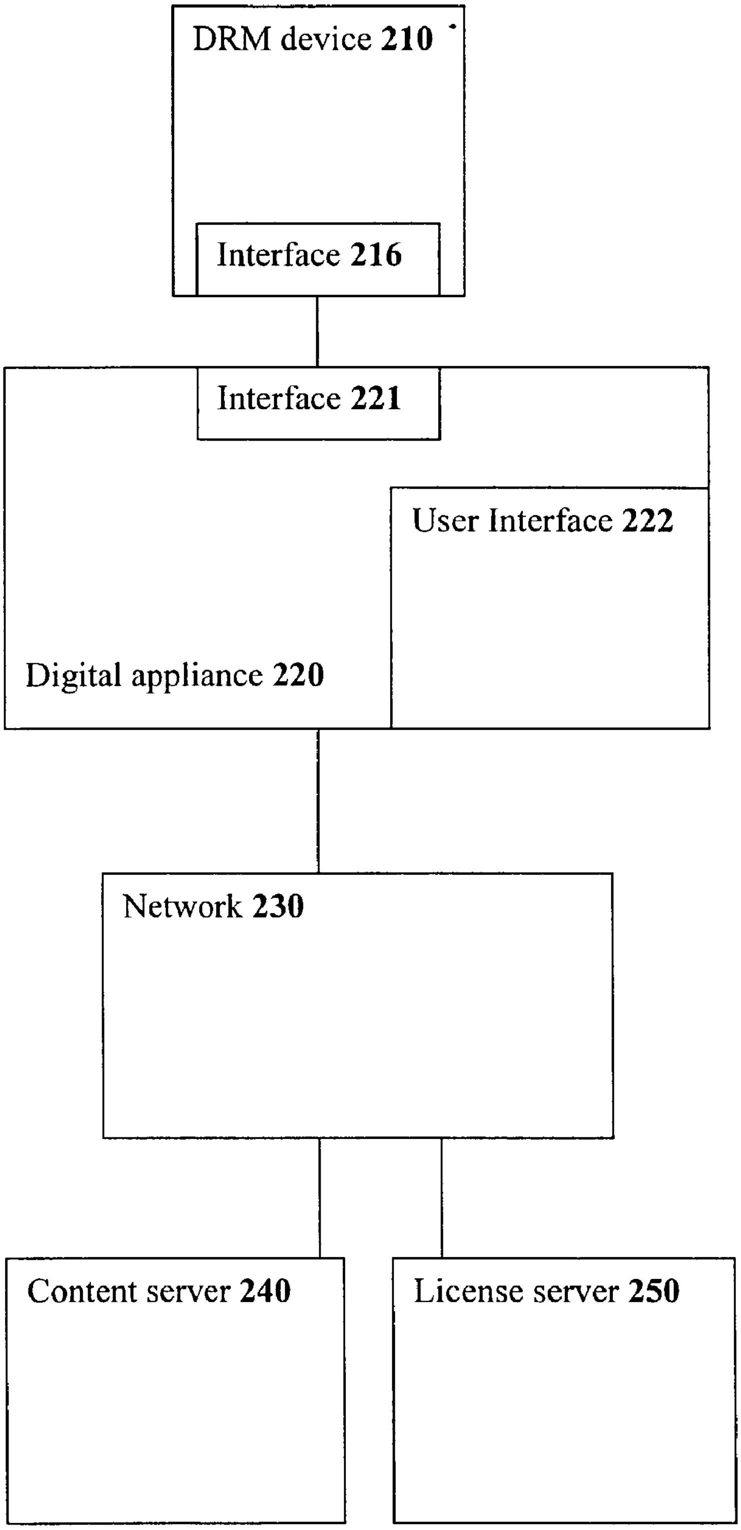

DRM device 210
Interface 216
Interface 221
User Interface 222
Digital appliance 220
Network 230
Content server 240
License server 250

FIG. 2

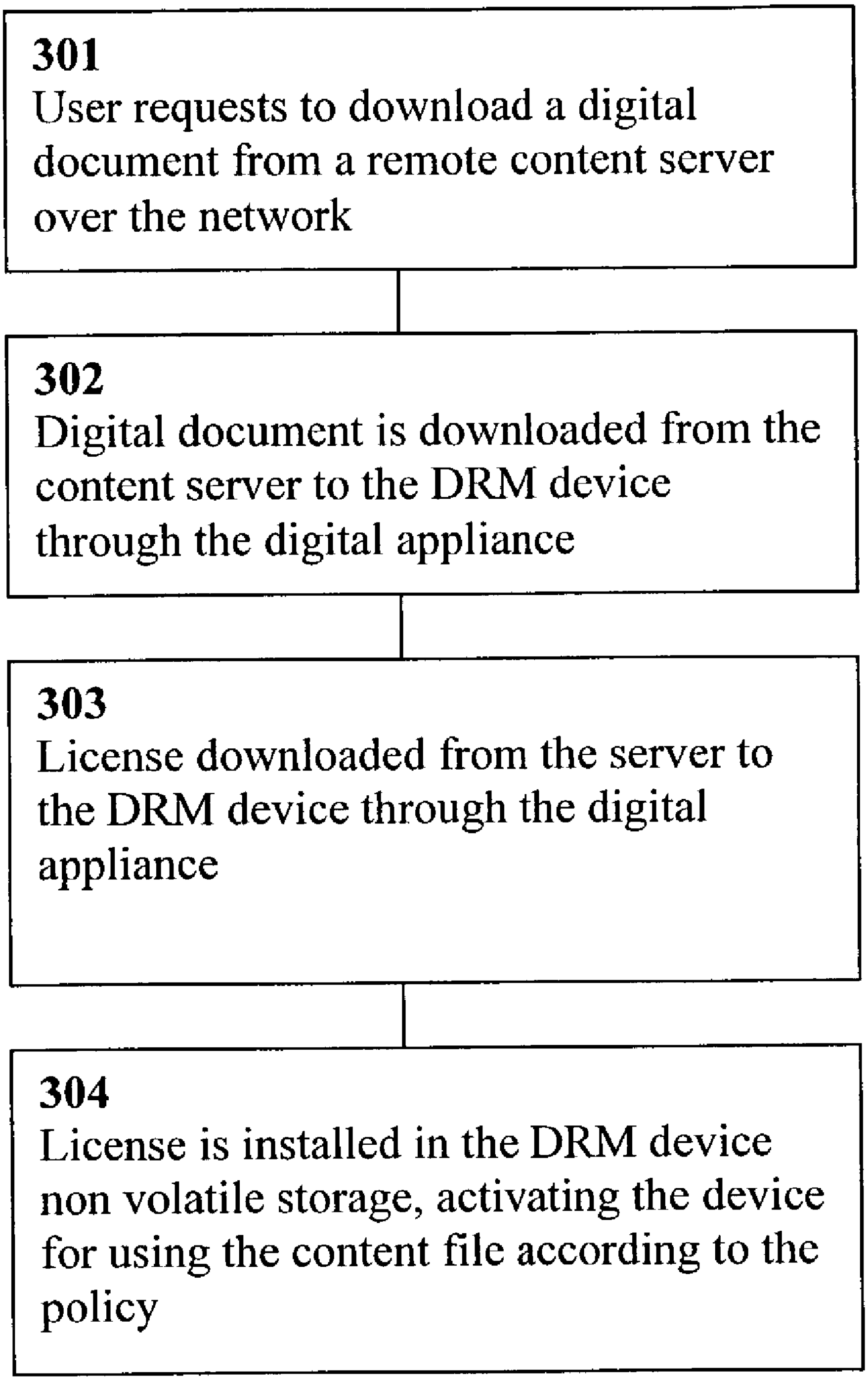

301
User requests to download a digital document from a remote content server over the network
302
Digital document is downloaded from the content server to the DRM device through the digital appliance
303
License downloaded from the server to the DRM device through the digital appliance
304
License is installed in the DRM device non volatile storage, activating the device for using the content file according to the policy

FIG. 3

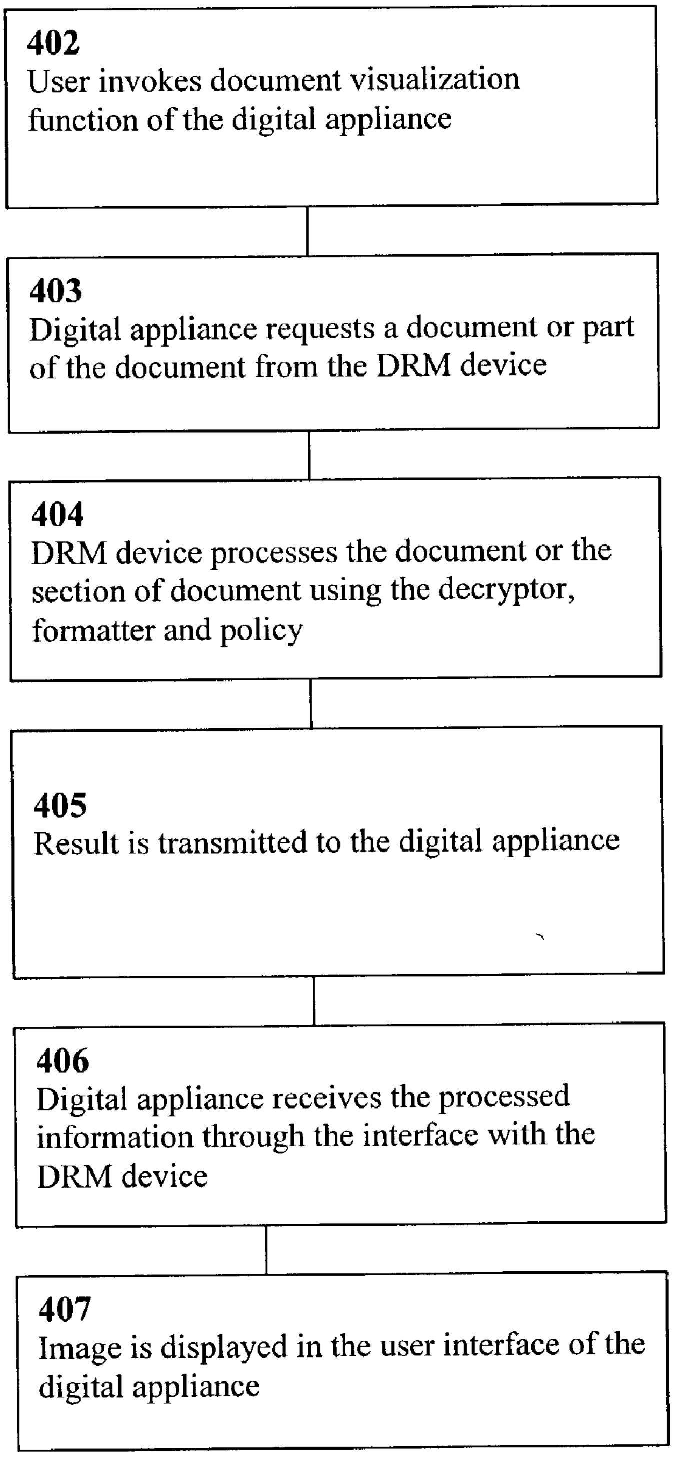

402
User invokes document visualization function of the digital appliance
403
Digital appliance requests a document or part of the document from the DRM device
404
DRM device processes the document or the section of document using the decryptor, formatter and policy
405
Result is transmitted to the digital appliance
406
Digital appliance receives the processed information through the interface with the DRM device
407
Image is displayed in the user interface of the digital appliance

FIG. 4

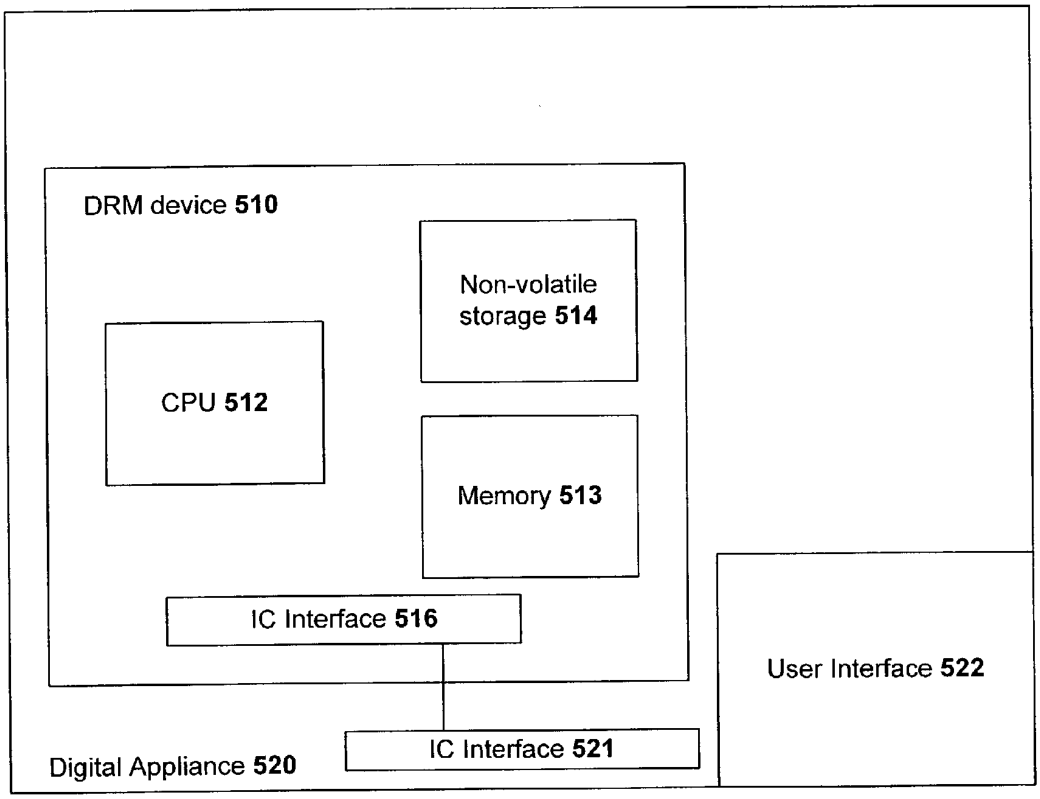

DRM device 510
Non-volatile storage 514
CPU 512
Memory 513
IC Interface 516
User Interface 522
Digital Appliance 520
IC Interface 521

FIG. 5

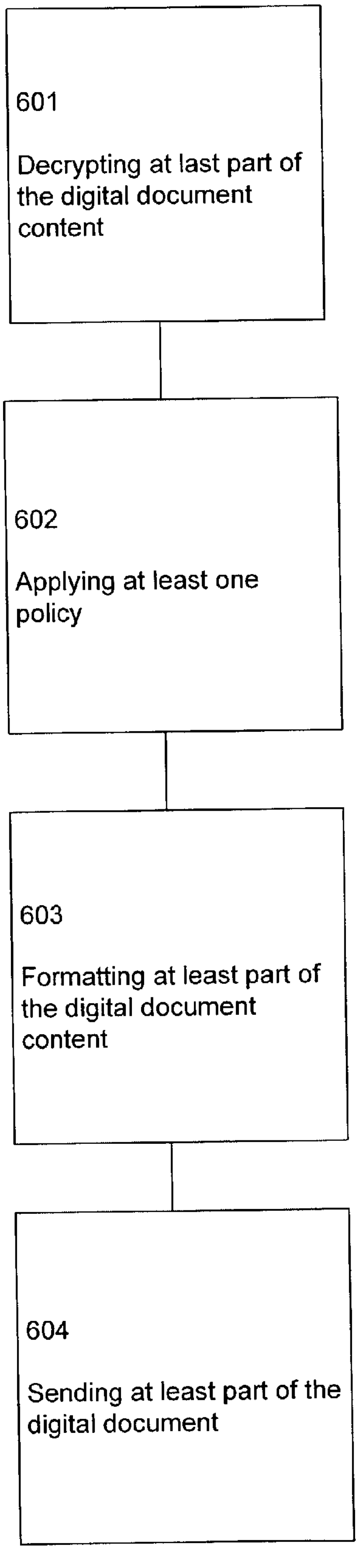

601
Decrypting at last part of the digital document content
602
Applying at least one policy
603
Formatting at least part of the digital document content
604
Sending at least part of the digital document

FIG. 6

APPARATUS, SYSTEM AND METHOD FOR SECURING DIGITAL DOCUMENTS IN A DIGITAL APPLIANCE

FIELD OF THE INVENTION

[0001] This invention generally relates to digital rights management. More particularly this invention relates to methods of securing digital documents to be used in a digital appliance such as a personal computer.

BACKGROUND OF THE INVENTION

[0002] The Internet worldwide network enables many digital appliances to interconnect and exchange information. A particular use of the Internet is to distribute digital files, specifically digital content such as digital books or music files, to the connected appliances.

[0003] The proliferation and distribution of digital music files is substantial. Various devices, programs and methods to listen to digital music are available, and an increasing number of music title exists in digital form. Unfortunately there exists a substantial amount of illegal copies of digital music files, such that the rights of the owner of the music cannot be exercised with respect to the illegal copies.

[0004] Digital books are substantially less popular and common than music. One of the reasons for the difference between the proliferation of music in digital form and books in digital form is the caution felt by book content rights owners against potential copyright infringement, a lesson learned from the experience of the music industry. Concerns about losing control over the management of rights prevents the usage of the Internet as a powerful digital content distribution infrastructure.

[0005] Digital rights management (DRM) systems are developed to challenge the above difficulties. Part of the function of a typical DRM system is to define the form of "rights-protected files"—methods that enable the use of digital files under limitations defined by the owner of the rights to the content. These systems typically involve cryptographic methods for the secure distribution of the content between a content repository or server and a digital appliance. Such methods typically require the appliance to include an implementation of cryptographic algorithms and hold cryptographic keys in order to gain access to the content. The access to the content is performed through a program that is DRM sensitive and is hereafter called—an electronic book reader.

[0006] Examples of electronic book reading software are the Adobe Acrobat, Adobe eBook Reader (http://www.adobe.com) and the Microsoft eBook Reader (http://www.microsoft.com/reader). Such software implements some form of DRM that is engaged when the users attempts to open and view a digital document. One of the operations performed by such electronic book readers is the process of decrypting the document using cryptographic methods and cryptographic keys. In order to do so, the reader program must have access to the cryptographic methods and keys; therefore the cryptographic methods and keys must reside within the access of the reader program. Typically the cryptographic methods, the keys, or both reside within the reader program, on the document itself, or somewhere within the appliance storage.

[0007] A digital appliance such as a computer is typically an open platform enabling computer programmers to develop programs. In some cases, software programs are developed for the purpose of hacking and locating the cryptographic keys and algorithms of a DRM system (hereafter referred to as hacking programs), in order to circumvent the DRM and gain illegal access to the content. This process is generally called an "attack" and if it succeeds it is commonly referred to as to "crack" the DRM system. A computer program that performs this function is referred hereafter as a hacking program.

[0008] Examples for such successful attacks are well known in the art. In late 2001, a programmer was able to crack the Microsoft eBook reader and locate the cryptographic methods and keys, producing a program that inputs an encrypted eBook file and outputs an illegal electronic book that is not protected (http://www.technologyreview.com/articles/innovation1101.asp). A similar cracking event of the Adobe system took place earlier that year (http://www.wired.com/news/politics/0,1283,45298,00.html).

[0009] Other forms of attacks include using programming tools. For example, software debuggers track and trap the electronic book information after the electronic book reader has decrypted it, retrieving the "protected" information. Such information includes the book text, images and attributes such as fonts, text color, and image locations, etc., which instruct the electronic book reader on how it should reconstruct the book for presentation to the user. A hacking program that cracks the reader and releases this information from the DRM system enables the construction of illegal copies of the original electronic book.

[0010] As a countermeasure, DRM systems have used more sophisticated cryptographic schemes and code obfuscation techniques. Other methods include adding tamper resistant hardware to store the cryptographic keys. Examples of such methods are cryptographic tokens such as iToken of Rainbow Technologies Inc. (http://www.rainbow.com/ikey/index.html) and using a smart card to store cryptographic keys and optionally cryptographic algorithms. Such solutions either reveal the cryptographic key to the digital appliance in the process of decrypting the information, or internally perform the cryptographic functions but reveal the end result in a raw form that can be hacked. In practice these methods were proven to slow, but not halt, an adversary. Given enough time and effort a computer program that "cracks" the DRM system may be written. It can be appreciated by those skilled in the art that such successful attacks may occur to such program readers that execute in an open environment that enables programmers to develop software programs. Similarly, cryptographic co-processors leave the content vulnerable after decryption.

[0011] Several ongoing initiatives focus on securing the personal computer itself.

[0012] As result, a major effort is being taken by the industry, led by companies such as Microsoft to protect some part of a personal computer by transforming that part into a closed system. (http://www.microsoft.com/presspass/features/2002/ju102/0724palladiumwp. asp). This initiative may produce a personal computer that is less sensitive to viruses, can be identified by service providers over the network, and can be used to build a DRM system. Microsoft's Palladium defines how to make the operating system of the personal computer secure. Once the operating system is secure, the PC is considered trusted and it can be used for purposes such as DRM. The Wave Embassy verification system secures an appliance. Unfortunately these initiatives will be realized only in future digital appliances, which must incorporate technology specific to Palladium and Wave Embassy for securing the personal computer itself. There is clearly an unmet need for a system, method and device for securing digital documents in a digital appliance.

SUMMARY OF THE INVENTION

[**0013**] The above-mentioned disadvantages and problems are addressed by the present invention, which will be understood by reading the following specification. To protect the cryptographic keys and cryptographic methods from being located within a digital appliance, according to the present invention the keys and methods are stored and executed in a dedicated DRM device that has processing capability distinct from the digital appliance, and does not provide an open environment for at least some security functions of the DRM device for programmers to develop programs. In some embodiments, another party may develop additional functions.

[**0014**] According to some embodiments, a digital document file or a section of the digital document that is protected is downloaded from an Internet server to the DRM device through a digital appliance. According to other embodiments, the digital document is already resident in the DRM device. Once the document is internal to the device, several processes may take place.

[**0015**] In some embodiments, if the document is in an encrypted form it is processed through a decryptor to produce a decrypted form. A decrypted digital document can be processed by a formatter internal to the DRM device to produce a formatted form of the digital document or the section of the digital document, such as, but not limited to, a bitmap image of a page of the document. Other examples of formatting include passing plain text to the digital appliance.

[**0016**] The DRM device can further process policies such as allowing or disallowing a formatted form of the document to be transferred to the digital appliance, for example in order to be presented to the user. The policy may be based on rights of use, time, number of usage events and so on. Some embodiments involve end use of digital documents. Other embodiments involve end use of music data and/or video data.

BRIEF DESCRIPTION OF THE DRAWINGS

[**0017**] The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of an embodiment of the invention with reference to the drawings, wherein:

[**0018**] **FIG. 1** is a schematic block diagram of an embodiment of the DRM device;

[**0019**] **FIG. 2** is a schematic block diagram of an exemplary system;

[**0020**] **FIG. 3** is a flowchart of an exemplary method for delivering a digital document file using the system of **FIG. 2**;

[**0021**] **FIG. 4** is a flowchart of an exemplary method for using the digital document file of **FIG. 3**;

[**0022**] **FIG. 5** is a schematic block diagram of another exemplary system; and

[**0023**] **FIG. 6** is a flowchart of another exemplary method for using the digital document file of **FIG. 3**.

DETAILED DESCRIPTION OF THE INVENTION

[**0024**] In the following detailed description of exemplary embodiments of the invention, reference is made to the drawings that illustrate specific exemplary embodiments in which the invention may be practiced. Those skilled in the art will appreciate that other embodiments may be utilized without departing from the spirit of the present invention, therefore the following detailed description of the invention should not be taken in a limiting sense. The scope of the invention is defined only by the appended claims.

[**0025**] **FIG. 1** is a diagram of an exemplary embodiment of the DRM device hardware **110**, which includes a central processing unit (CPU) **112**, an optional system memory **113**, non-volatile storage **114**, and an interface **116** to connect the device **110** to a digital appliance **120**. There may be only one or a plurality of central processing units **112**, as there may optionally be only one or a plurality of system memory **113** or non-volatile storage **114**. There may be only one or a plurality of interfaces **116**; the invention is not so limited. The non-volatile storage **114** may be included in the CPU **112** or be discrete from the CPU **112**; generally, components or subcomponents of the DRM device hardware **110** may be combined with other components or subcomponents of the DRM device for higher integration and perhaps lower cost.

[**0026**] The CPU **112** may be a general purpose CPU or a CPU with dedicated functions. Furthermore the CPU **112** may include internal memory, and internal non-volatile storage which in the description of the present invention may serve a similar purpose of the system memory **113**, and non-volatile storage **14** respectively. The CPU **112**, the non-volatile storage **114**, and/or other components may be implemented as a tamper resistant hardware, or sections of the CPU **112**, the non-volatile storage **114**, and/or other components may be tamper resistant; the invention is not so limited.

[**0027**] The non-volatile storage **114** may be any of several types of storage including semiconductor based media such as read only memory (ROM), electronic erasable programmable read only memory (EEPROM), flash memory or battery backed up random access memory (RAM); or magnetic media storage such as hard disk drive or floppy disk, or the like.

[**0028**] The interface **116** can connect the DRM device **110** with a digital appliance **120** in both physical and communication aspects. The physical aspect can be, for example directly, through one or more cables, and/or wireless. The communication aspect of the interface **116** allows data exchange between the DRM device and the digital appliance. The interface **116** may be any of several types of interfaces, for example PCI, ISA, Universal Serial Bus (USB), FireWire, IDE, SCSI, RS-232 or other serial interface, parallel interface, Compact Flash (CF) interface, Sony Memory Stick interface, Multimedia Card (MMC), secure digital (SD), Bluetooth, Infiniband, and/or any other type of interface that may be used to connect a DRM device with a digital appliance.

[**0029**] The digital appliance **120** is used by an end user for some end use of one or more digital documents. A digital document is data which has an end use of being read by an end user, and at some point prior to end use is stored and/or represented in numerical form. The digital document can have various purposes, for example a corporate purpose such as a sales presentation, a legal contract, a finance spreadsheet, or the like; or an academic purpose, such as an academic book, a published paper, a student class pack reader, or the like; or a commercial purpose, for example a newspaper, a periodical journal, a comics journal, or the like; or the like various purposes that a digital document may have. The digital appliance **120** may be one of several digital appliances such as a personal computer, tablet computers, personal digital assistant (PDA) or other types of hand held devices, cell phones, programmable consumer electronics and the like. End use includes use of the DRM device by an end user to access digital document content. Some examples of tasks which can be performed in connection with accessing the document content include viewing the content of the document or a section of the document, modifying the document, searching the document for a text string, copying parts or all of the document, selecting text within the document to perform an operation on that text, add overlay comments on top of existing content, respond to assignments by adding content to the document or adding content to a matching but separate document, listening to a voice version of the document, printing sections or all of the document, sharing the document with other end users, transferring all or part of the document to other end users, transferring the rights to use the document to other end users, aggregation of several documents or sections of several documents into one or more new documents and other like operations that a user may apply to a digital document. The invention is not so limited.

[**0030**] The non-volatile storage **114** contains instructions which may be executed by the CPU **112**. The non-volatile storage **114** further may contain: an optional unique device serial number, a method of authentication such as a unique pair of public and private cryptographic keys and a signed authenticity certificate. The instructions stored in the non volatile storage **114** allow the digital appliance **120** to access a portion of the non volatile storage **114** through the interface **116**, but prevent access to another portion of the non volatile storage **114**, including a portion that stores the private cryptographic key and a portion that stores instructions that execute in a closed environment without enabling user access. The non-volatile storage may also store a plurality of methods for authentication; the invention is not so limited.

[**0031**] **FIG. 2** is a diagram of an exemplary embodiment of the system which includes a DRM device **210** with an interface **216**, a digital appliance **220** with an interface **221** which matches the interface **216** of the DRM device **210**, a user interface component **222** on which a processed document may be presented (for example as a visual image, synthesized audio or other form) to the user, the network **230**, a content server **240** which is a computer that can transfer digital documents over the network and a license server **250** which is a computer that may transfer authentication and/or decryption and/or policy and/or formatting information over the network. According to one embodiment that information is embedded in one or more files. According to one embodiment the servers are optionally interconnected. The system may include a plurality of DRM devices **210**, digital appliances **220**, content servers **240** and license servers **250**, the invention is not so limited. It may be appreciated by those skilled in the art that the content server **240** and the license server **250** may be implemented as separate or unite hardware and/or software components.

[**0032**] The interface **221** connects the digital appliance **220** with a DRM device **210**. The interface **221** may be any of several types that may be used to connect a device with a digital appliance. The interface **221** of the digital appliance matches the type of interface **216** of the DRM device in a form that enables information to pass between the DRM device **210** and the digital appliance **220**.

[**0033**] The content server **240** is a computer that can be accessed through a network **230** such as the Internet network. The content server **240** can respond to requests to download content such as digital electronic documents. Examples of content servers can be Amazon.com or another on-line bookseller web site that enables downloading of electronic books to a personal computer, a university web site that enables downloading of electronic versions of articles to a researcher's personal computer, and a corporate web site that enables employees to download corporate documents to their personal computers. A license server **250** is a computer that can be accessed through a network **230** such as the Internet network. A license server **250** can respond to requests to download information such as authentication and/or decryption and/or policy and/or formatting information. This data may include: definition of policies to be used by the DRM device policies, definition of formatting to be used by the DRM device formatters, definition of decryption to be used by the DRM device decryptors, definition of authentication to be used by the DRM device authenticators, parts of the text of the electronic document or parts of the electronic document, information regarding the user, information regarding the rights of the user to one or more end uses (the user may have access to all possible end uses or less than all possible end uses) of the document or part of the document, information regarding the vendor/owner/operator of the system, information regarding the specific DRM device, and other information. The information may be utilized by the DRM device or the digital appliance while the user makes use of the content or in preparation to enable the user to make use of the content or any additional information. According to one embodiment the content server **240** and the license server **250** are implemented as separate entities that interconnect through a network and do not directly interconnect. According to another embodiment the servers directly interconnect. According to another embodiment the content server **240** and the license server **250** are implemented as a single entity. The invention is not so limited.

[**0034**] An authenticator implemented in a DRM device participates in the process of authenticating the DRM device to a remote server over a network. An authenticator may implement one of several methods of authentication including sending a device ID number to the remote server. Another authenticator uses an encryption secret key known only to the device and the server, and bases the authentication on challenging the device in order to verify that it has possession of the secret key. In an exemplary embodiment of such an authentication process the server sends an encrypted message to the device, and the authenticator at least decrypts the message and returns it to the server. In some embodiments, the same key can be used in a variety of methods to authenticate, for example, by signing a plaintext message and/or decrypting an encrypted message. In some embodiments, the authenticator responds to challenges by performing a series of operations such as decrypt a message, process the result, encrypt the result, and return it to the server for verification. For this authentication process to occur, the secret key may be stored in the device prior to the authentication process. The stored key can be a single key stored equally on all devices or a dedicated key unique to each device. In the latter case the server should know in advance which key is stored within which device. Another method to authenticate uses a public and private key and a digital certificate. In such an embodiment, the authenticator has access to a private key and a matching public key stored in the device The private key must be kept secret, but the public key may be made public. The server may then challenge the authenticator with a message encrypted with the device public key to ensure it has access to the matching private key. In some embodiments, the authenticator signs a message but does not necessarily encrypt the message. Optionally the server can receive from the device a digital certificate, which contains device identification information such as the device serial number or device ID and/or the public key of the device and/or additional information relating to the device, the server, the organization operating the system or any other information. The device identification information is digitally signed by a trusted authority, such as the vendor of the device, owner of the server, the organization operating the system and/or another trusted authority to form a digital certificate for that device. Some embodiments of the authenticator can authenticate the DRM device and/or a user of the DRM device.

[**0035**] A decryptor in the device participates in the process of transforming encrypted documents or sections of documents into a decrypted form. A decryptor may implement one or more of several methods: symmetric algorithms such as DES, 3DES, AES, and IDEA; and/or asymmetric algorithms such as RSA, Diffie-Hellman, elliptic curve; and/or others. A decryptor may implement one or a plurality of decryption methods. A decryptor may include hashing algorithms such as DSA, MD2, MD4, MD5, HMAC and/or SHA1 and/or others to retrieve a signature and check origin and integrity of the data received. The decryption key or plurality of decryption keys for such operations may originate in one or a plurality of sources. For example, decryption key data can be stored in the non-volatile storage of the DRM device, received from the digital appliance, and/or received from a network server, such as through the digital appliance. Some embodiments receive digital document content which is at least partly decrypted. In such embodiments, obviously the decryptor may or may not process the already decrypted portion. The decryptor can at least partly decrypt—for example, fully decrypt part of a document, and/or perform one or more decryption steps, which can be the complete decryption process or a subset of the complete decryption process, for a whole or part of the document. In some embodiments, the document can be received at least partly as plaintext—in other words, received as at least partly unencrypted.

[**0036**] A policy in the device participates in the process of verifying the eligibility of end use of a document or a section of a document, allowing or disallowing operations such as decrypting, formatting, searching, and/or transmitting an output to the digital appliance. The verification may check one or several eligibility options, including the right to use the document, the right to use the document up to a certain date, the right to use the document between certain dates, the right to use the document after a certain date, the right to use the document for a certain accumulated usage time, the right to use the document for a certain number of times, the right to transfer the document, the right to modify the document, the right to add overlay information on the document, the right to save the document into the device and/or another location, the right to save the overlay information into the device and/or another location, the right to copy the document, the right to copy portions of the document, the right to copy specific sections of the document, and other rights related to an end user in connection with an end use of the document. These might be checked by the policy to produce a result that might be one or more possible actions such as allowing the output to be transmitted to the digital appliance, disallowing the output from being transmitted to the digital appliance, erasing the document or part of the document, and/or allowing or disallowing operations such as search, cut, paste, copy, edit, save, and other operations that a user may perform while in an end use of the document.

[**0037**] A formatter defines a process step in formatting a document into a presentable form. A formatter may do one or more formatting operations including: selecting the section of the document to be presented; conversion of the text, graphics and images to a single or set of digital images in one of many formats such as a bitmap image (BMP) or like form or compressed image such as JPEG, TIFF, GIF; or any other like form; setting spaces between characters and letters according to the required display form; searching the text for a particular text string; generating the layout of the document; drawing the text characters in the appropriate font and font size; and other operations performed in the preparation and conversion of a document into a presentable form. Some embodiments of a formatter degrade at least part of the document. Some embodiments arrange a presentation of the digital document content by presenting visual and/or audio information, such as presenting a voice version of the document.

[**0038**] **FIG. 3** is a flow chart describing an exemplary sequence of operations carried out when a user downloads content from a network server. In step **301** one or more users request a digital document to be downloaded to the DRM device that is connected to the digital appliance. Typically following step **301**, the server drives a phase of proving the eligibility of the user to receive the document. User eligibility to receive the content is determined by the server, following rules such as payment, free for use, user authentication, registration or other similar methods that may be used by a user to prove eligibility or to become eligible to receive the document. Once the server is ready to download the content, it sends the content through the network to the digital appliance that is attached to the network. The content may be encrypted or parts of it may be encrypted. According to one embodiment the DRM device must be presently attached to the digital appliance at the time of transmission. According to another embodiment the DRM device does not necessarily have to be attached at the time of transmission of the document and can be made present later when the document is to be used. At step **302** the document is transmitted from the network server (depicted as content server) to the digital appliance and from the digital appliance to the DRM device. According to one embodiment the document is completely transferred to the digital appliance before being transferred to the DRM device. According to another embodiment the document is transferred in sections, where each section is transferred to the DRM device at its own pace. On step **303** the license is transferred from the network server (depicted as license server) to the digital appliance and from the digital appliance to the DRM device. The license can be one or more files. The license contains information used by the policy, authenticator, decryptor, and/or formatter in the DRM device. According to another embodiment the license server and the content sever are implemented as a single server. According to another embodiment the license is embedded in the document to form a single file transferred from a single server. It may be appreciated by those skilled in the art that there exist other methods to sequence the transfer process with the result of having the document or part of the document and the license transferred to the DRM device. Step **304** describes the installation of the license in the non-volatile storage of the DRM device. Once installed in the DRM device, the license may activate the usage of the document according to the rights defined in the license. According to one embodiment the activation is performed immediately following the installation process. According to another embodiment the activation is performed in a later timeframe, such as at the time of usage of the document.

[**0039**] **FIG. 4** is a flow chart describing an exemplary sequence of operations for using a document for visualization. In step **402** the user invokes a document usage function in the digital appliance. In step **403** the digital appliance further sends requests to the DRM device. In step **404** the DRM device processes the request by performing a sequence of operations, optionally involving one or more decryptors, one or more formatters, and one or more policies on the document or part of the document, before transferring the result to the digital appliance in step **405**. According to another embodiment, part or all of the operations that involve the decryptors, formatters and/or policies is performed before the request from the digital appliance is received. The order of the operations of the decryptors, formatters and policies can be altered and executed in any sequence. The invention is not so limited.

[**0040**] **FIG. 4** is a flow chart describing an exemplary sequence of operations for using a document for visualization. In step **402** the user invokes a document usage function in the digital appliance. In step **403** the digital appliance further sends requests to the DRM device. In step **404** the DRM device processes the request by performing a sequence of operations, optionally involving one or more decryptors, one or more formatters, and one or more policies on the document or part of the document, before transferring the result to the digital appliance in step **405**. According to another embodiment, part or all of the operations that involve the decryptors, formatters and/or policies is performed before the request from the digital appliance is received. The order and existence of the operations of the decryptors, formatters and policies can be altered and can occur in any sequence. The invention is not so limited.

[**0041**] **FIG. 5** is a diagram of another exemplary embodiment of the system which includes a DRM device **510** with an IC interface **516**, a digital appliance **520** with an IC interface **521** which matches the IC interface **516** of the DRM device **510**, and a user interface component **522** on which a processed document may be presented (for example as a visual image, synthesized audio or other form) to the user. One example of the DRM device **510** is an integrated circuit executing instructions. The DRM device **510** can be included in the digital appliance **520**. In some embodiments the code or data can be stored inside the non-volatile storage of the DRM device IC, and/or can be in storage external to the DRM device IC. The DRM device IC can execute independently from a processor of the digital appliance.

[**0042**] **FIG. 6** is a flow chart describing another exemplary sequence of operations for using a document for visualization. In step **601**, at least part of the digital document content is decrypted. In step **602**, at least one policy is applied. In step **603**; at least part of the digital document content is formatted. In step **604**, at least part of the digital document is sent. The order and existence of the operations can be altered and can occur in any sequence.

1. A digital rights management (DRM) device for digital content management, the DRM device adapted to be coupled to a digital appliance for end use of at least part of a digital document, the DRM device comprising:

one or more nonvolatile storages adapted to store:

1) one or more authenticators;

2) one or more decryptors, wherein at least one of the decryptors is adapted to at least partly decrypt at least part of the digital document;

3) one or more policies, wherein at least one of the policies at least partly controls access to at least part of the digital document; and

4) one or more formatters, wherein at least one of the formatters at least partly arranges a presentation of at least part of the digital document; and

one or more interfaces coupled to at least one of the nonvolatile storages, the one or more interfaces for receiving and sending at least part of the digital document,

wherein at least part of the sent digital document is sent to the digital appliance for end use of the digital document.

2. The device of claim 1, wherein the DRM device is coupled to the digital appliance.

3. The device of claim 2, wherein the digital appliance is a computer.

4. The device of claim 2, wherein the digital appliance is a personal digital assistant.

5. The device of claim 2, wherein the digital appliance is a mobile phone.

6. The device of claim 1, wherein the digital document is an electronic book.

7. The device of claim 1, wherein the digital document is a corporate document

8. The device of claim 1, wherein the digital document is an academic document.

9. The device of claim 1, wherein the digital document is a commercial document.

10. The device of claim 1, wherein the DRM device, prior to a first attempted end use of the DRM device, has stored in the nonvolatile storage at least one of: at least one of the one or more authenticators, at least one of the one or more decryptors, at least one of the one or more policies, and at least one of the one or more formatters.

11. The device of claim 1, wherein the DRM device, at least partly responsive to a first attempted use of the DRM device, downloads into the nonvolatile storage at least one of: at least one of the one or more authenticators, at least one of the one or more decryptors, at least one of the one or more policies, and at least one of the one or more formatters.

12. The device of claim 1, wherein the DRM device couples to the digital appliance via a physical connection to the digital appliance.

13. The device of claim 12, wherein the physical connection includes one or more cables.

14. The device of claim 1, wherein the DRM device couples to the digital appliance by directly physically connecting to the digital appliance.

15. The device of claim 1, wherein the DRM device couples to the digital appliance by remotely connecting to the digital appliance.

16. The device of claim 1, wherein the DRM device couples to the digital appliance by wirelessly connecting to the digital appliance.

17. The device of claim 1, wherein at least part of the received digital document content is received encrypted.

18. The device of claim 1, wherein at least part of the received digital document content is received as plaintext.

19. The device of claim 1, wherein the presentation is at least partly visual data.

20. The device of claim 1, wherein the presentation is at least partly audio data.

21. A digital rights management (DRM) integrated circuit (IC) for digital content management, the DRM IC adapted to be included in a digital appliance for end use of at least part of a digital document, the DRM IC comprising:

one or more nonvolatile storages adapted to execute:

1) one or more authenticators;

2) one or more decryptors, wherein at least one of the decryptors is adapted to at least partly decrypt at least part of the digital document;

3) one or more policies, wherein at least one of the policies at least partly controls access to at least part of the digital document; and

4) one or more formatters, wherein at least one of the formatters at least partly arranges a presentation of at least part of the digital document; and

one or more interfaces coupled to at least one of the nonvolatile storages, the one or more interfaces for receiving and sending at least part of the digital document,

wherein at least part of the sent digital document is sent to a processor in the digital appliance for end use of the digital document.

22. A digital rights management (DRM) system for digital content management, the DRM system including:

one or more servers, wherein the one or more servers send at least one of: authentication data, decryption data, policy data, formatting data, and at least part of digital document content data;

a digital appliance for end use of at least part of the digital document content; and

a DRM device adapted to be coupled to the digital appliance, the DRM device comprising:

one or more nonvolatile storages adapted to store:

1) one or more authenticators;

2) one or more decryptors, wherein at least one of the decryptors is adapted to at least partly decrypt at least part of the digital document content;

3) one or more policies, wherein at least one of the policies at least partly controls access to at least part of the digital document content; and

4) one or more formatters, wherein at least one of the formatters at least partly arranges a presentation of at least part of the digital document content; and

one or more interfaces coupled to at least one of the nonvolatile storages, the one or more interfaces for receiving at least data from the one or more servers and sending at least part of the digital document content to at least the digital appliance,

wherein at least part of the sent digital document content is sent to the digital appliance for end use of at least part of the digital document content.

23. A method of digital rights management (DRM) with a DRM device for at least part of the digital document content, comprising:

applying at least one policy to at least part of the digital document content in the DRM device;

formatting at least part of the digital document content in the DRM device; and

sending at least part of the digital document for end use to a digital appliance coupled to the DRM device.

24. The device of claim 23, further comprising:

decrypting at least part of the digital document content in the DRM device.

* * * * *

EXHIBIT 11

US009059969B2

(12) **United States Patent**
McNulty

(10) **Patent No.: US 9,059,969 B2**
(45) **Date of Patent: *Jun. 16, 2015**

(54) **APPARATUS, METHOD AND SYSTEM FOR A TUNNELING CLIENT ACCESS POINT**

(71) Applicant: **Scott McNulty**, Rowayton, CT (US)

(72) Inventor: **Scott McNulty**, Rowayton, CT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 18 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/960,514**

(22) Filed: **Aug. 6, 2013**

(65) **Prior Publication Data**

US 2014/0172958 A1 Jun. 19, 2014

Related U.S. Application Data

(63) Continuation of application No. 12/950,321, filed on Nov. 19, 2010, now Pat. No. 8,539,047, which is a continuation of application No. 10/807,731, filed on Mar. 23, 2004, now Pat. No. 7,861,006.

(51) **Int. Cl.**

G06F 15/16	(2006.01)
G06F 15/177	(2006.01)
H04L 29/08	(2006.01)
H04L 9/32	(2006.01)
G06F 13/00	(2006.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/04*** (2013.01); ***H04L 63/0272*** (2013.01); ***H04L 63/0428*** (2013.01); ***H04L 9/3226*** (2013.01); ***H04L 9/3247*** (2013.01); *H04L 2209/56* (2013.01); *H04L 2209/76* (2013.01); *H04L 2209/80* (2013.01)

(58) **Field of Classification Search**

CPC H04L 2209/56; H04L 2209/76; H04L 2209/80; H04L 63/0272; H04L 63/0428; H04L 67/04; H04L 9/3226; H04L 9/3247

USPC 709/203, 250; 713/150; 711/115

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,960,085	A	9/1999	de la Huerga	
6,098,097	A *	8/2000	Dean et al.	709/220
6,134,662	A *	10/2000	Levy et al.	726/11
6,199,108	B1 *	3/2001	Casey et al.	709/220
6,547,130	B1	4/2003	Shen	
6,763,399	B2 *	7/2004	Margalit et al.	710/13
6,799,077	B1 *	9/2004	Hauet	700/2
6,928,463	B1 *	8/2005	Tene et al.	709/203
7,051,157	B2 *	5/2006	James	711/115

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1168137	1/2002

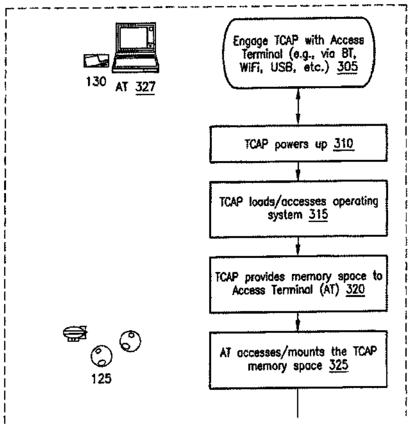
Primary Examiner — Alina N Boutah

(74) *Attorney, Agent, or Firm* — Locke Lord LLP

(57) **ABSTRACT**

The disclosure details the implementation of a tunneling client access point (TCAP) that is a highly secure, portable, power efficient storage and data processing device. The TCAP "tunnels" data through an access terminal's (AT) input/output facilities. In one embodiment, the TCAP connects to an AT and a user employs the AT's user input peripherals for input, and views the TCAP's activities on the AT's display. This enables the user to observe data stored on the TCAP without it being resident on the AT, which can be useful to maintain higher levels of data security. Also, the TCAP may tunnel data through an AT across a communications network to access remote servers. The disclosure also teaches a plug-n-play virtual private network (VPN).

29 Claims, 17 Drawing Sheets


130
AT 327
Engage TCAP with Access Terminal (e.g., via BT, WiFi, USB, etc.) 305
TCAP powers up 310
TCAP loads/accesses operating system 315
TCAP provides memory space to Access Terminal (AT) 320
AT accesses/mounts the TCAP memory space 325
125

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,178,724	B2	2/2007	Tamagno et al.	
7,213,766	B2 *	5/2007	Ryan et al.	235/492
7,308,584	B2	12/2007	Himmel et al.	
7,546,340	B2 *	6/2009	Terasawa	709/203
7,549,161	B2	6/2009	Poo et al.	
7,558,953	B2 *	7/2009	Osthoff et al.	713/161
7,762,470	B2 *	7/2010	Finn et al.	235/492
2002/0044663	A1 *	4/2002	King et al.	380/284
2002/0073340	A1	6/2002	Mambakkam et al.	
2002/0184349	A1 *	12/2002	Manukyan	709/221
2002/0194499	A1	12/2002	Audebert et al.	
2003/0005337	A1	1/2003	Poo et al.	
2003/0028649	A1 *	2/2003	Uhlik et al.	709/228
2003/0158891	A1 *	8/2003	Lei et al.	709/203
2003/0182456	A1	9/2003	Lin et al.	
2004/0044897	A1	3/2004	Lim	
2004/0127254	A1 *	7/2004	Chang	455/557
2005/0172075	A1 *	8/2005	Marcus	711/115
2005/0197859	A1 *	9/2005	Wilson et al.	705/2
2005/0198221	A1 *	9/2005	Manchester et al.	709/220
2006/0052085	A1 *	3/2006	Rodriguez et al.	455/411
2006/0071066	A1 *	4/2006	Vanzini et al.	235/380
2006/0294249	A1	12/2006	Oshima et al.	
2007/0038870	A1	2/2007	Ciesinger	
2007/0274291	A1 *	11/2007	Diomelli	370/352
2008/0233942	A9 *	9/2008	Kim	455/419

* cited by examiner

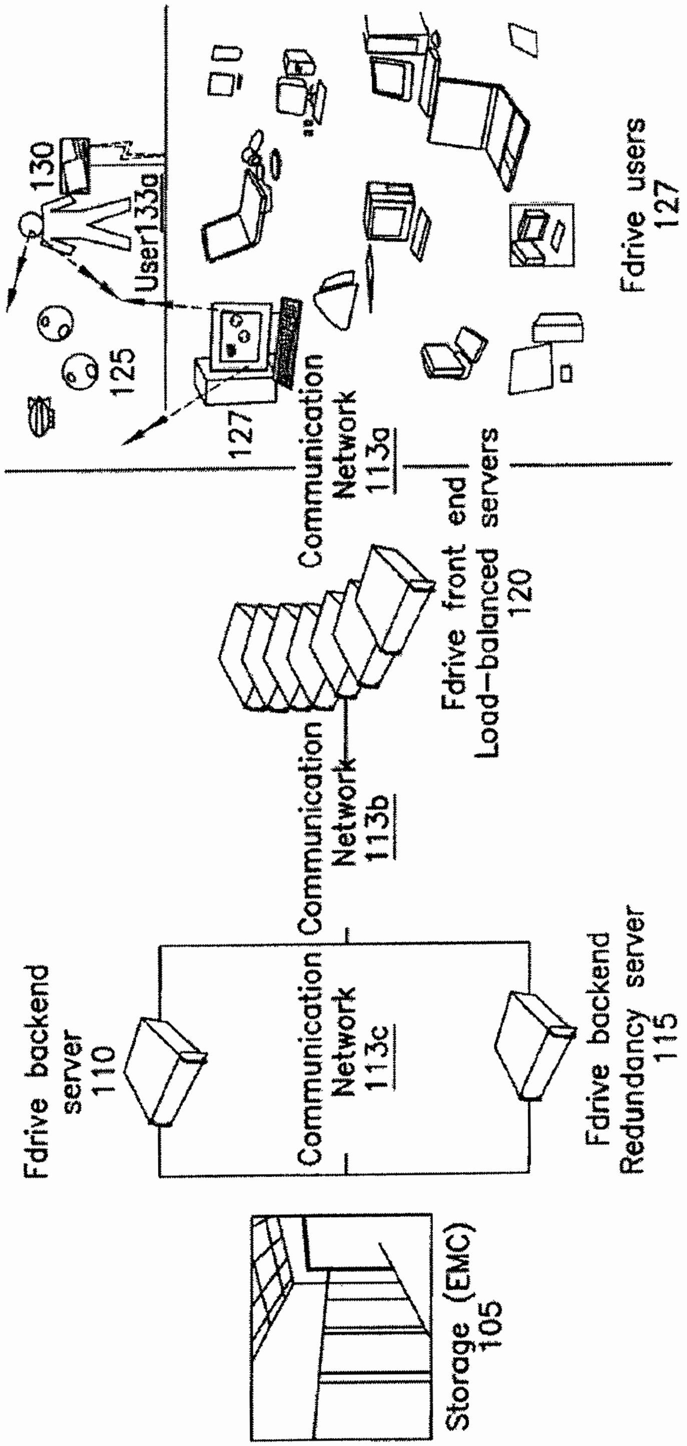

Storage (EMC)
105
Fdrive backend
server
110
Communication
Network
113c
Fdrive backend
Redundancy server
115
Communication
Network
113b
Fdrive front end
Load-balanced servers
120
Communication
Network
113a
127
125
User133a
130
Fdrive users
127

Fig. 1

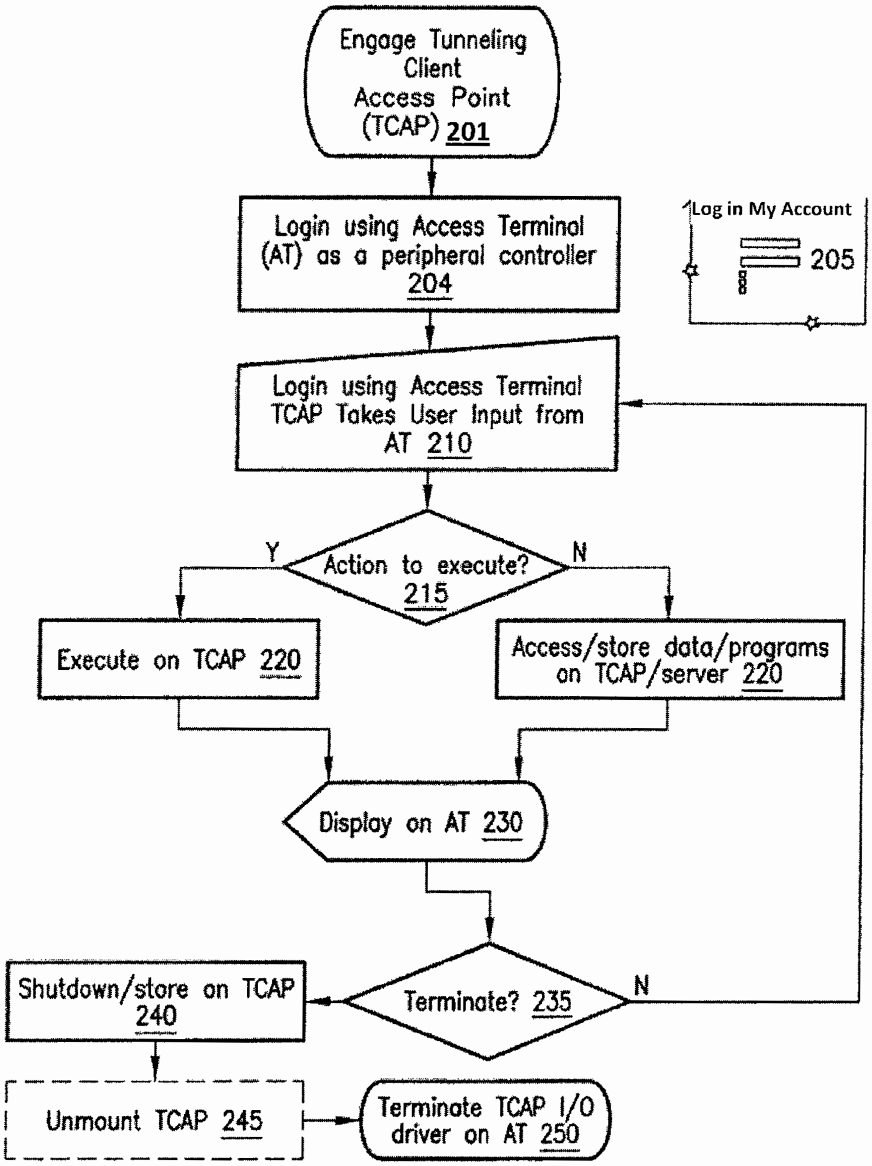

Engage Tunneling Client Access Point (TCAP) 201
Login using Access Terminal (AT) as a peripheral controller 204
Log in My Account
205
Login using Access Terminal TCAP Takes User Input from AT 210
Action to execute? 215
Y
N
Execute on TCAP 220
Access/store data/programs on TCAP/server 220
Display on AT 230
Terminate? 235
N
Shutdown/store on TCAP 240
Unmount TCAP 245
Terminate TCAP I/O driver on AT 250

Fig. 2

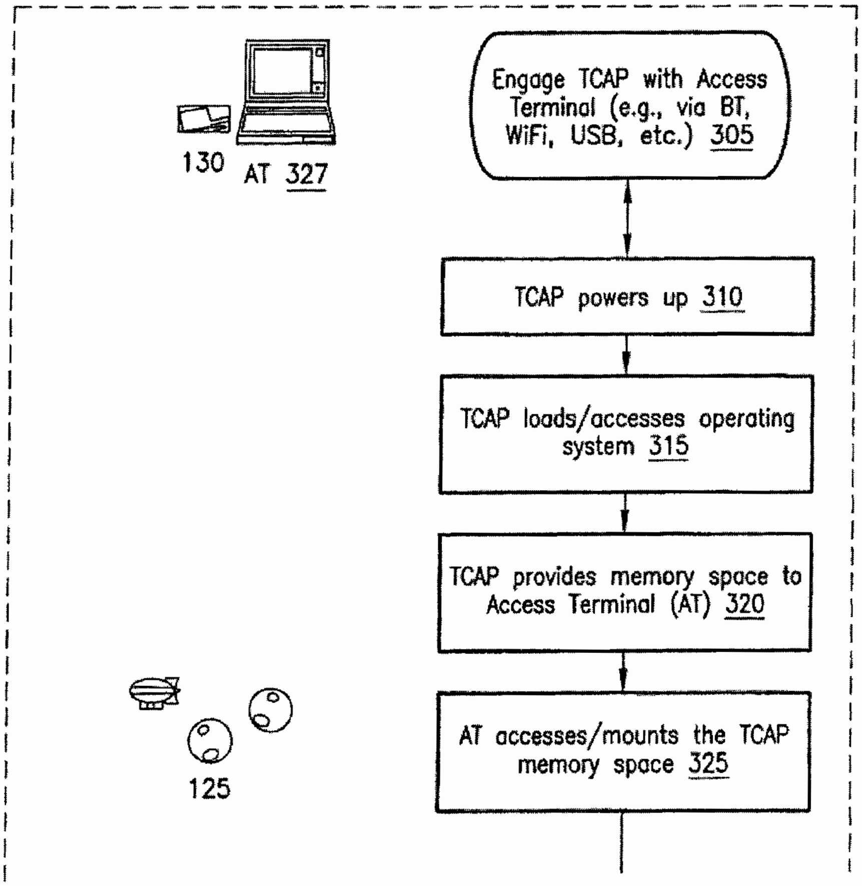

130
AT 327
Engage TCAP with Access Terminal (e.g., via BT, WiFi, USB, etc.) 305
TCAP powers up 310
TCAP loads/accesses operating system 315
TCAP provides memory space to Access Terminal (AT) 320
AT accesses/mounts the TCAP memory space 325
125

Fig. 3a

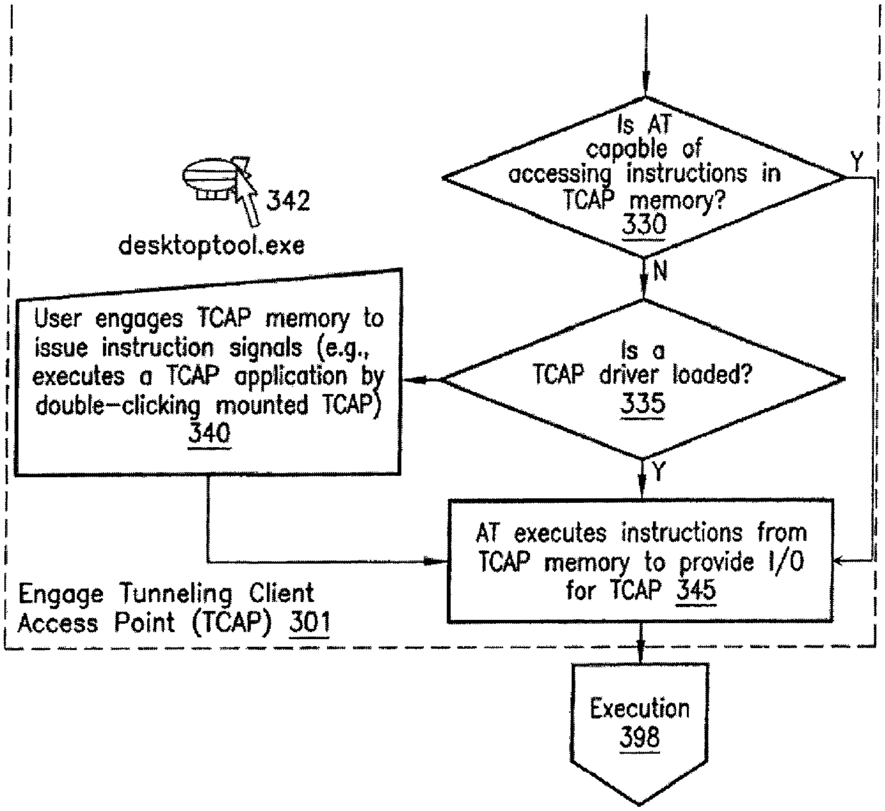

Is AT capable of accessing instructions in TCAP memory? 330
Y
N
342
desktoptool.exe
User engages TCAP memory to issue instruction signals (e.g., executes a TCAP application by double-clicking mounted TCAP) 340
Is a TCAP driver loaded? 335
Y
AT executes instructions from TCAP memory to provide I/O for TCAP 345
Engage Tunneling Client Access Point (TCAP) 301
Execution 398

Fig. 3b

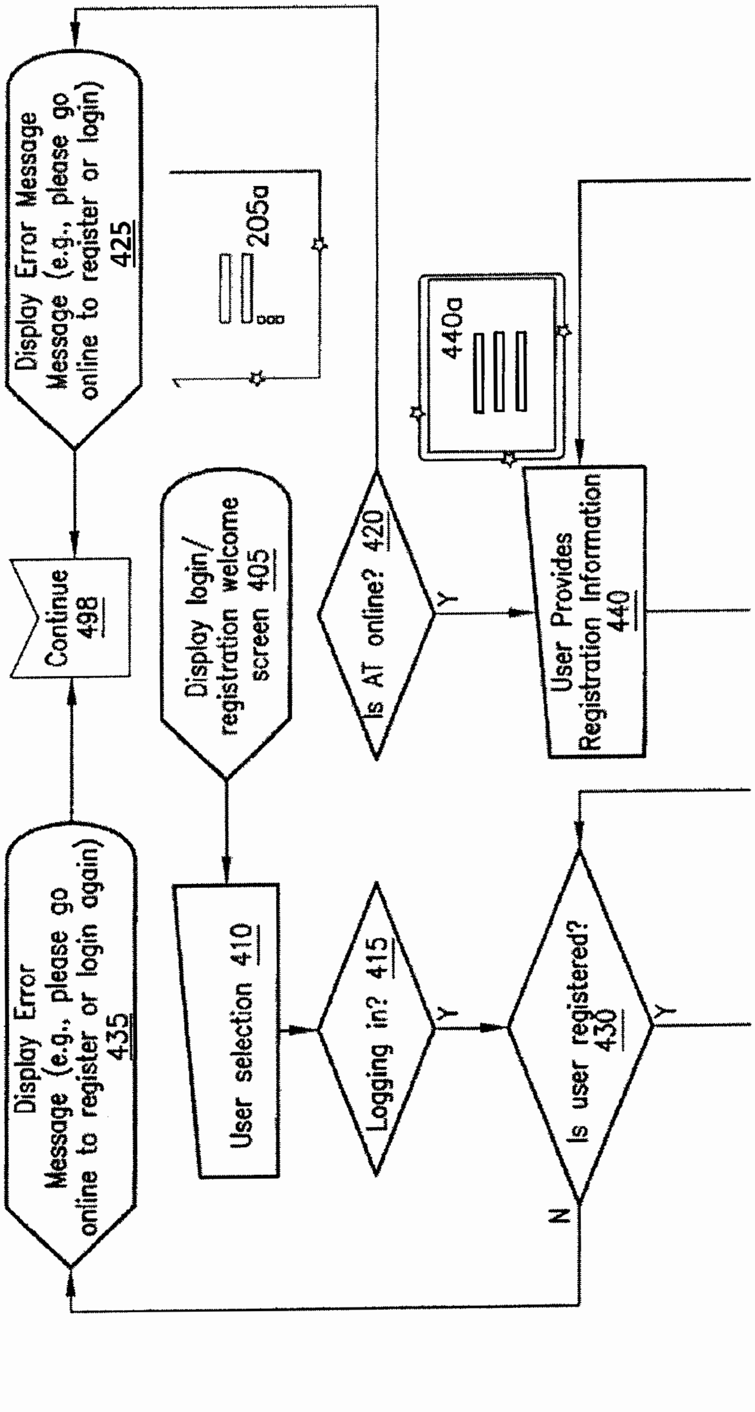

Display login/ registration welcome screen 405
User selection 410
Logging in? 415
Y
Is user registered? 430
Y
N
Display Error Message (e.g., please go online to register or login again) 435
Continue 498
Is AT online? 420
Y
User Provides Registration Information 440
440a
205a
Display Error Message Message (e.g., please go online to register or login) 425

Fig. 4a

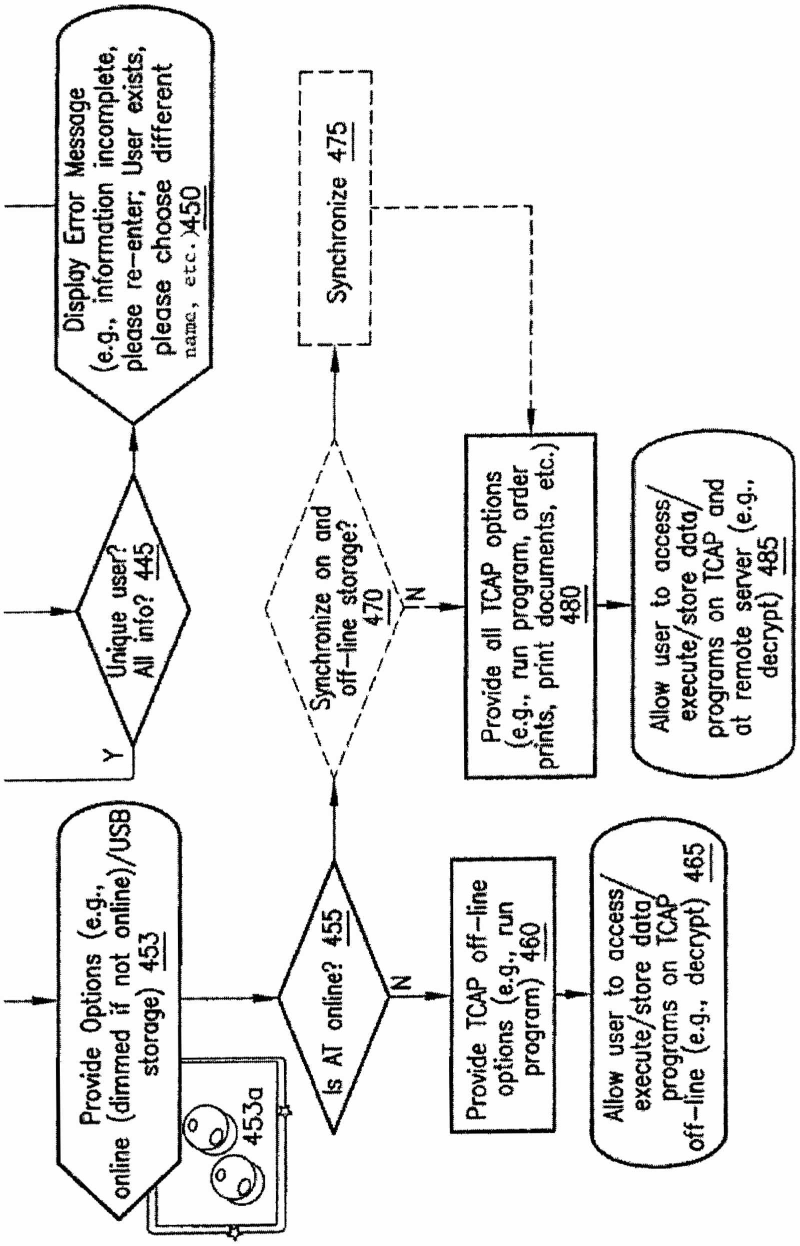

Display Error Message (e.g., information incomplete, please re-enter; User exists, please choose different name, etc.) 450
Synchronize 475
Unique user? All info? 445
Y
Synchronize on and off-line storage? 470
N
Provide all TCAP options (e.g., run program, order prints, print documents, etc.) 480
Allow user to access/ execute/store data/ programs on TCAP and at remote server (e.g., decrypt) 485
Provide Options (e.g., online (dimmed if not online)/USB storage) 453
453a
Is AT online? 455
N
Provide TCAP off-line options (e.g., run program) 460
Allow user to access/ execute/store data/ programs on TCAP off-line (e.g., decrypt) 465

Fig. 4b

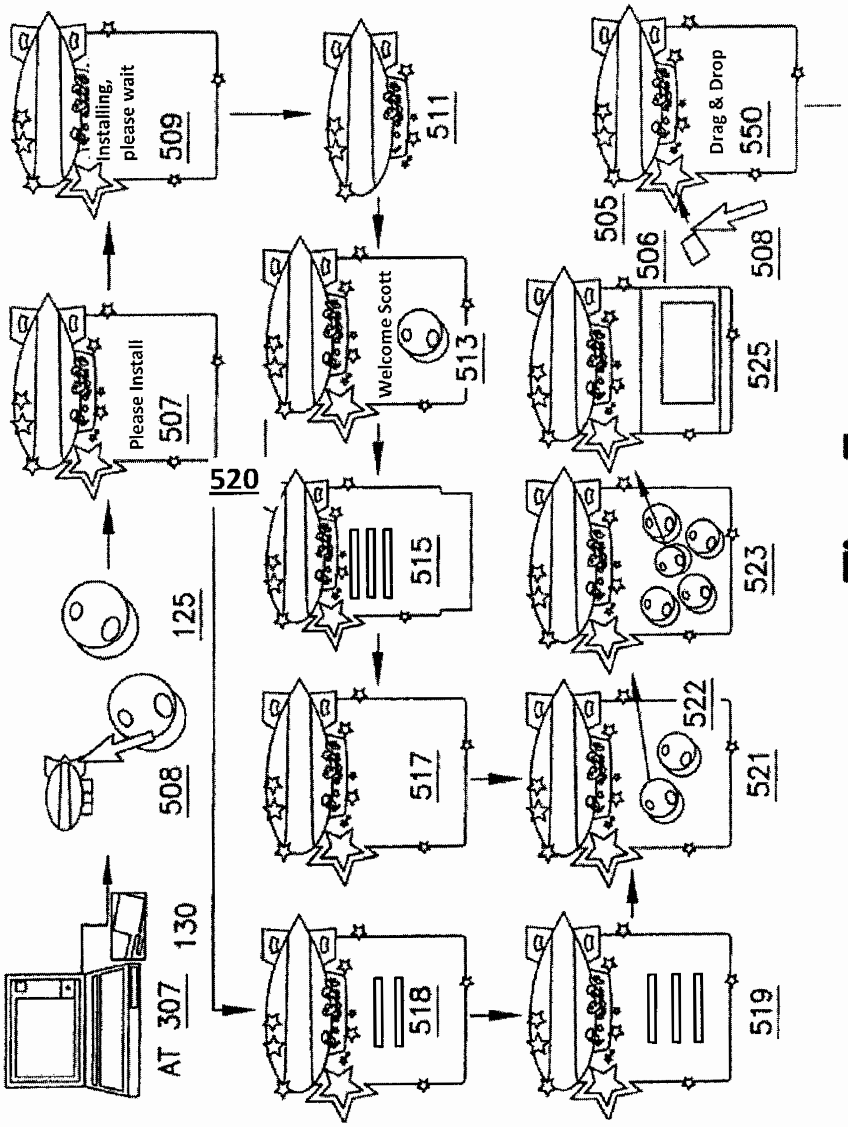

AT 307
130
508
125
Please Install
507
Installing,
please wait
509
511
520
Welcome Scott
513
515
517
518
519
521
522
523
525
505
506
508
Drag & Drop
550

Fig. 5a

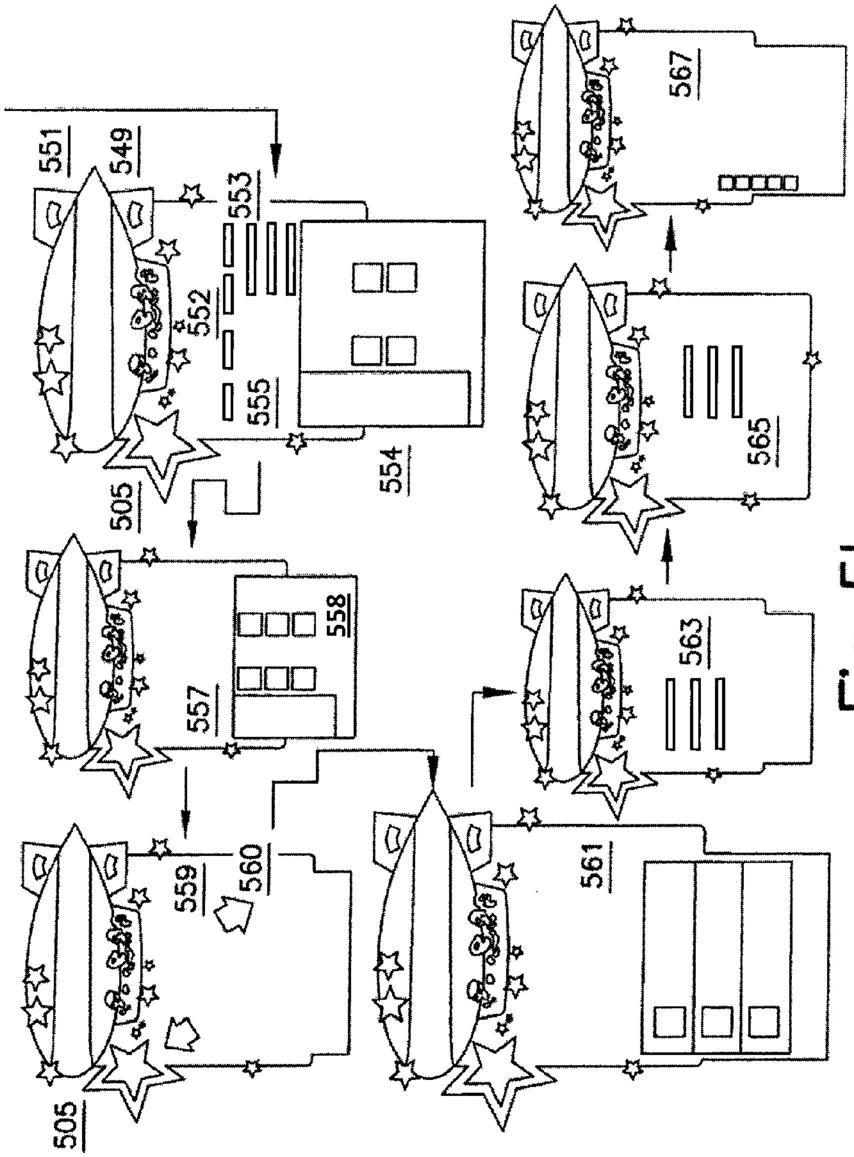


Fig. 5b

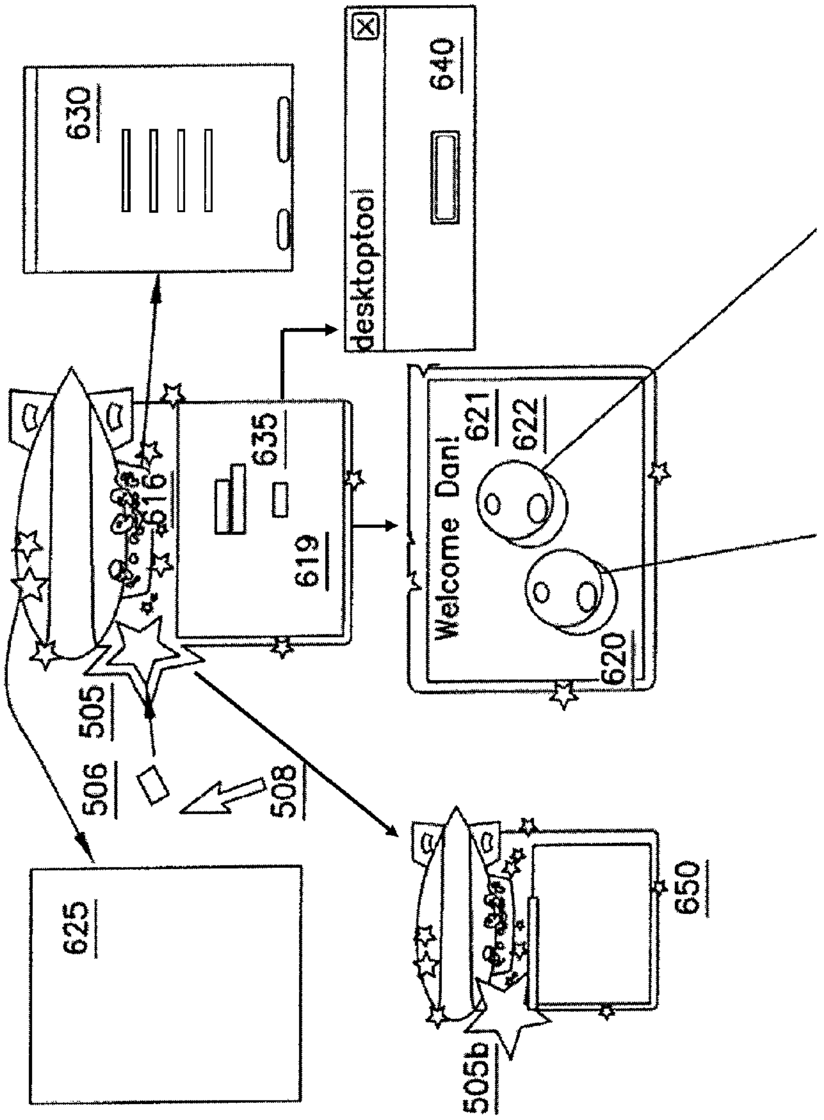

630
desktoptool
640
635
616
619
621
622
Welcome Dan!
620
505
506
508
625
505b
650

Fig. 6a

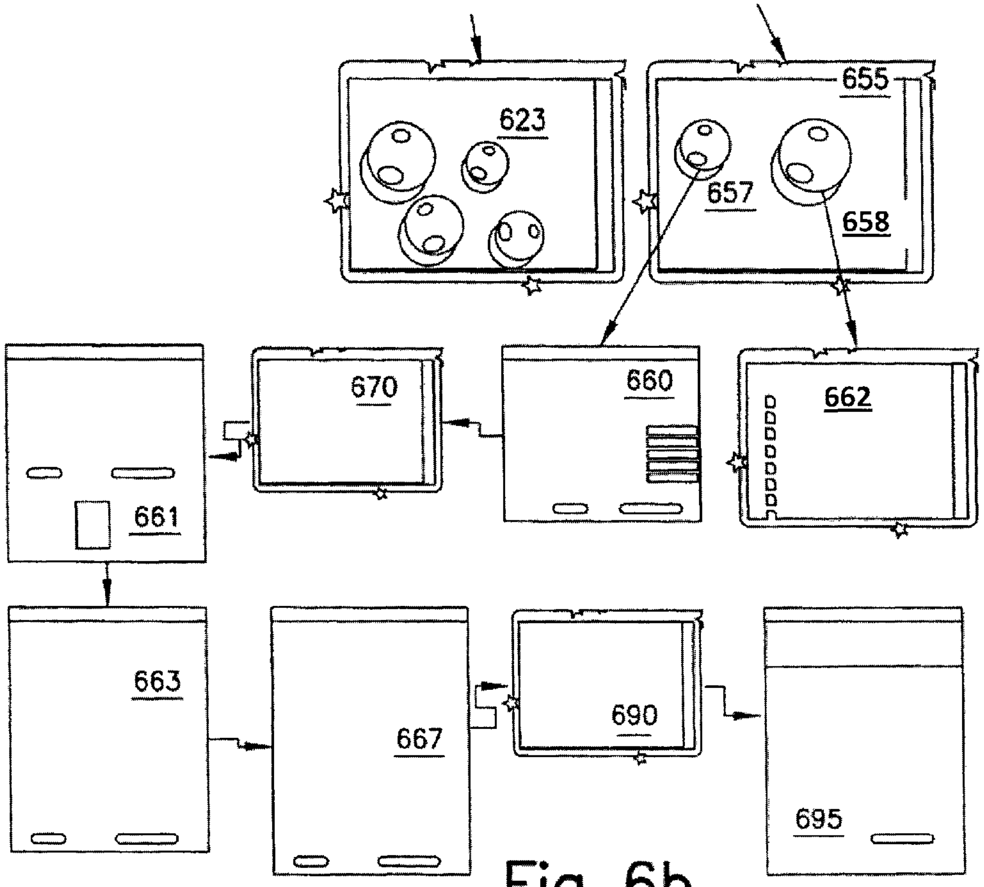


Fig. 6b

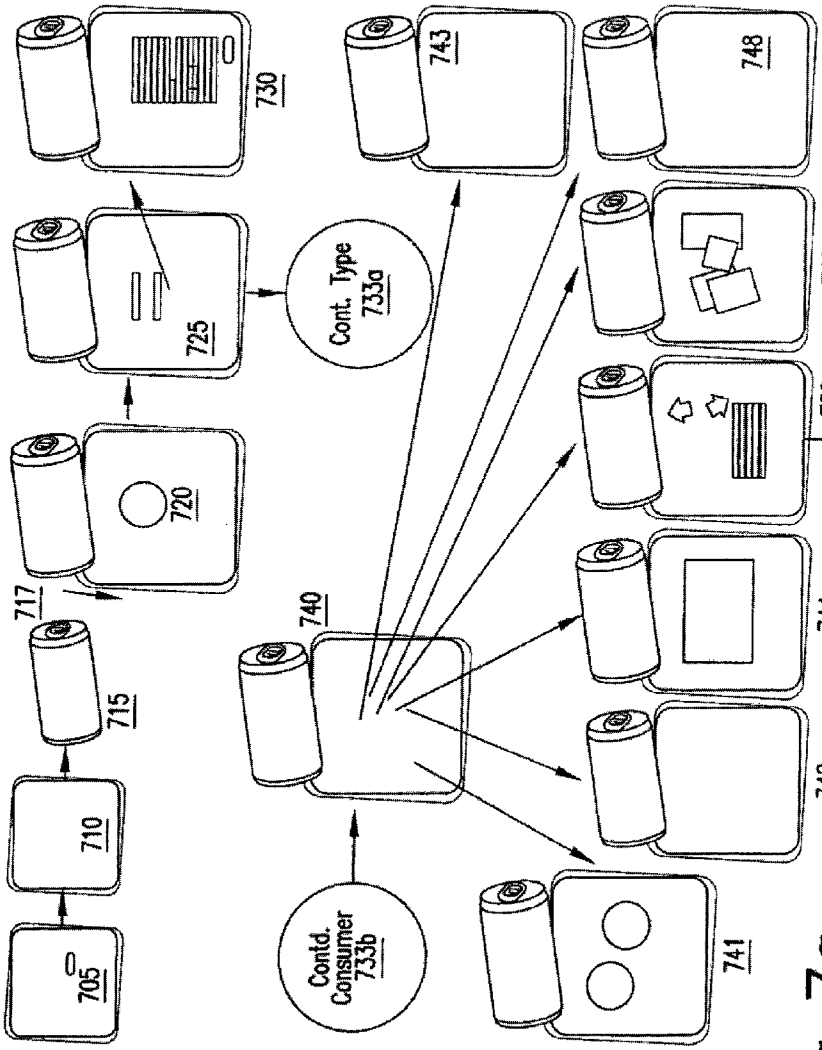

705
710
715
717
720
725
730
Cont. Type 733a
740
Contd. Consumer 733b
741
742
743
744
746
748
750

Fig. 7a

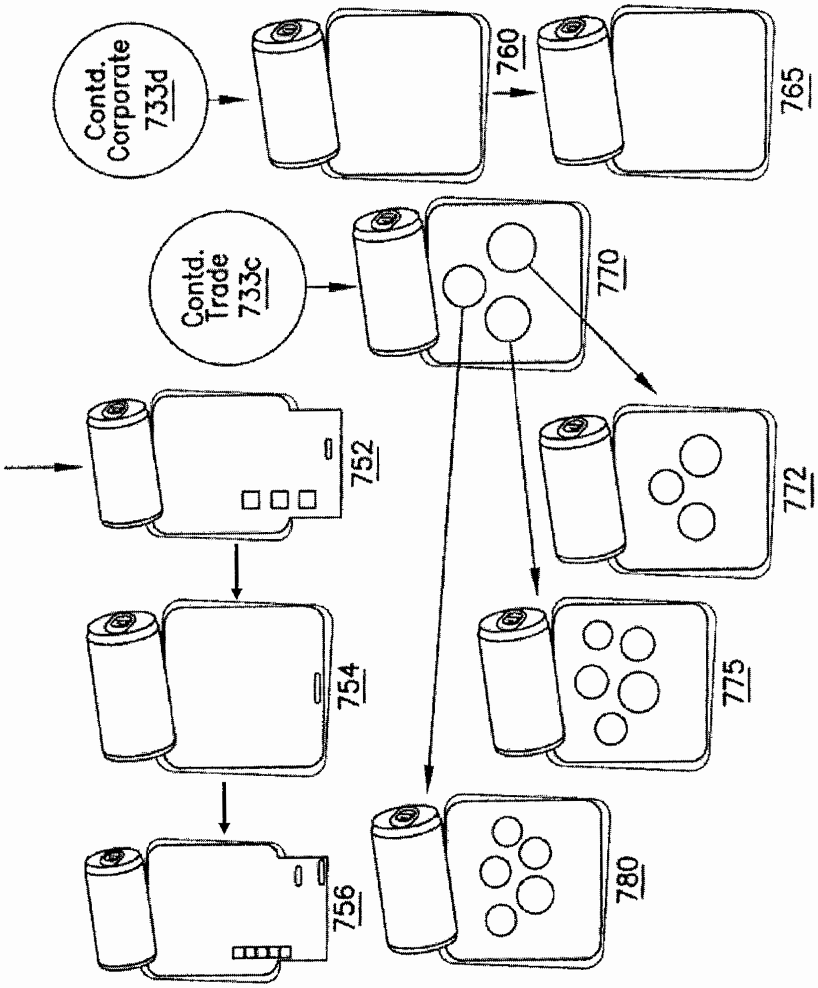

Contd.
Corporate
733d
760
765
Contd.
Trade
733c
770
752
772
754
775
756
780

Fig. 7b

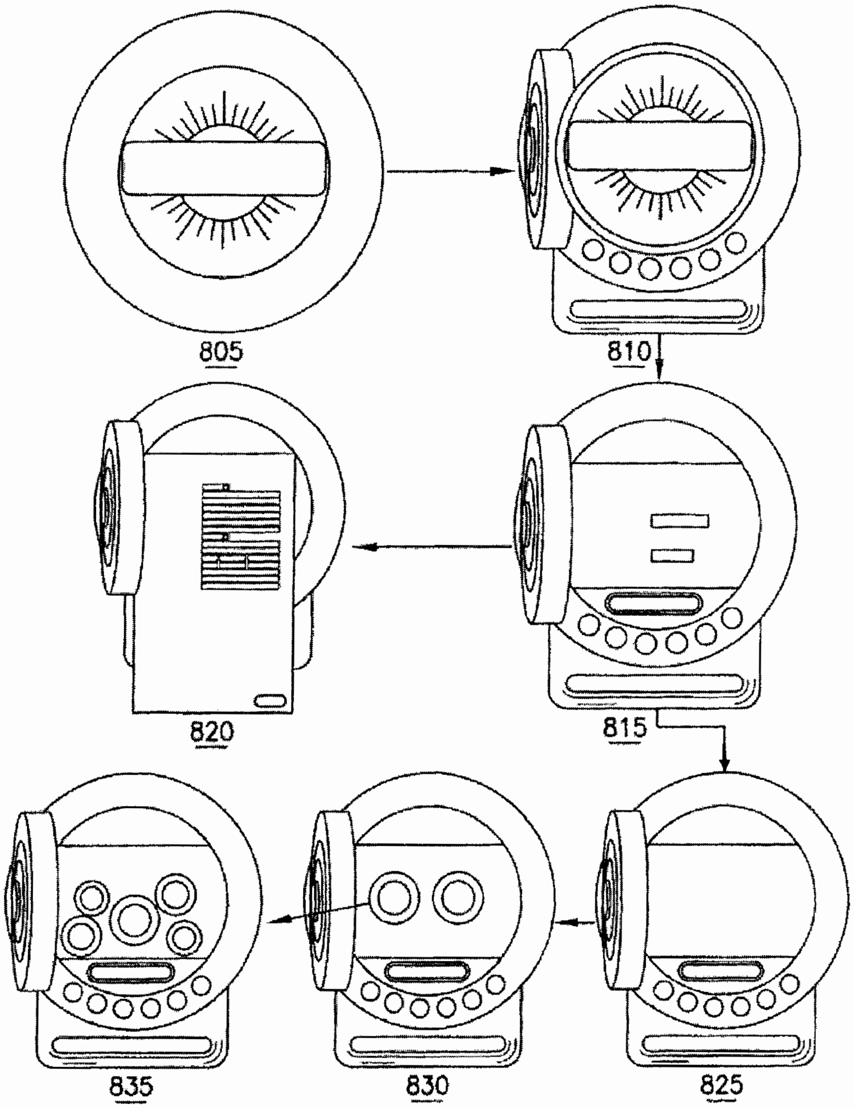


Fig. 8

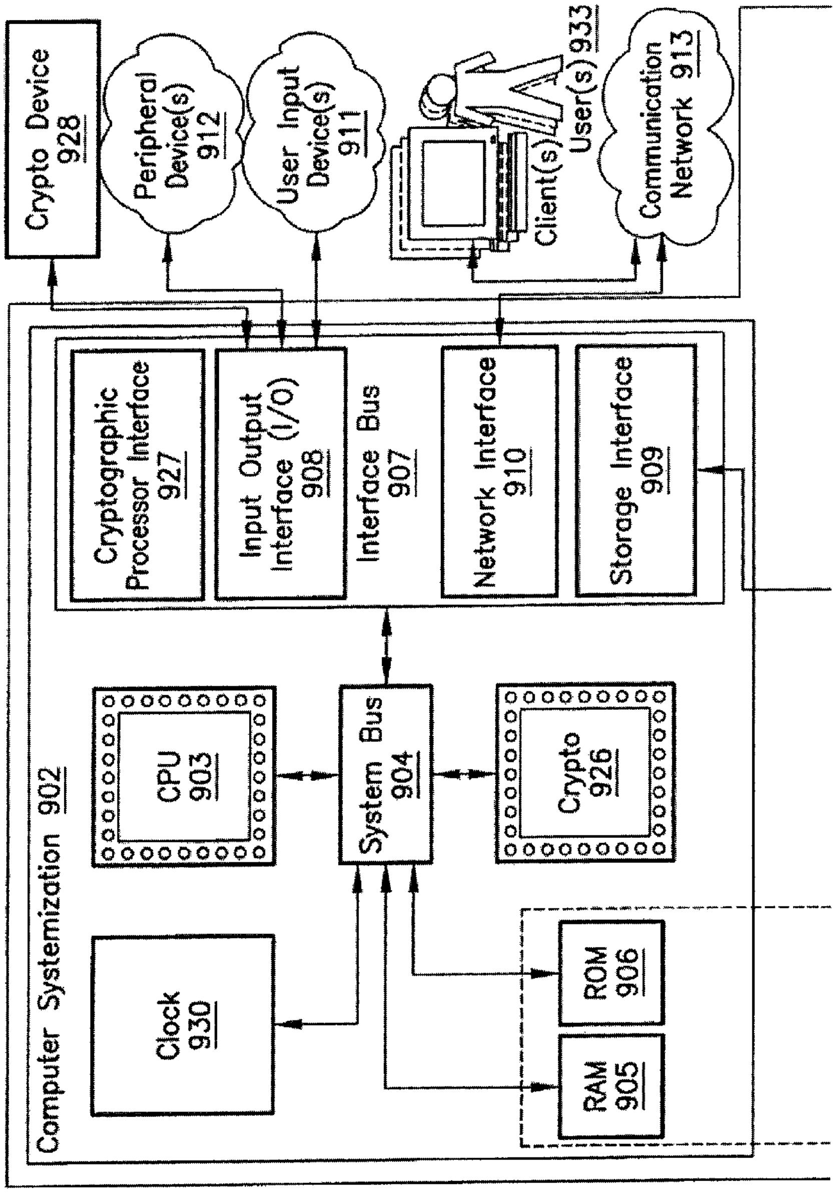

Crypto Device 928
Peripheral Device(s) 912
User Input Device(s) 911
Client(s)
User(s) 933
Communication Network 913
Computer Systemization 902
Cryptographic Processor Interface 927
Input Output Interface (I/O) 908
Interface Bus 907
Network Interface 910
Storage Interface 909
CPU 903
System Bus 904
Crypto 926
Clock 930
ROM 906
RAM 905

Fig. 9a

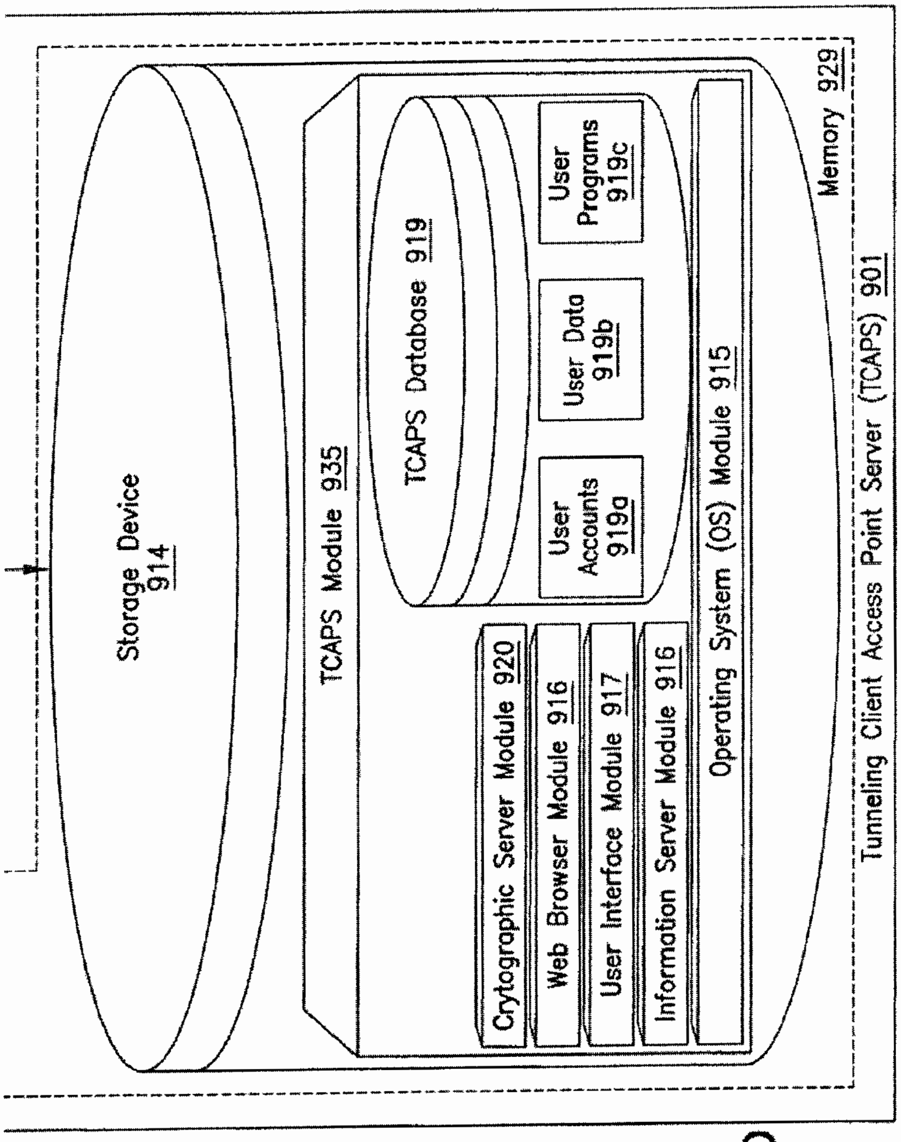

Storage Device 914
TCAPS Module 935
TCAPS Database 919
User Accounts 919a
User Data 919b
User Programs 919c
Crytographic Server Module 920
Web Browser Module 916
User Interface Module 917
Information Server Module 916
Operating System (OS) Module 915
Memory 929
Tunneling Client Access Point Server (TCAPS) 901

Fig.9b

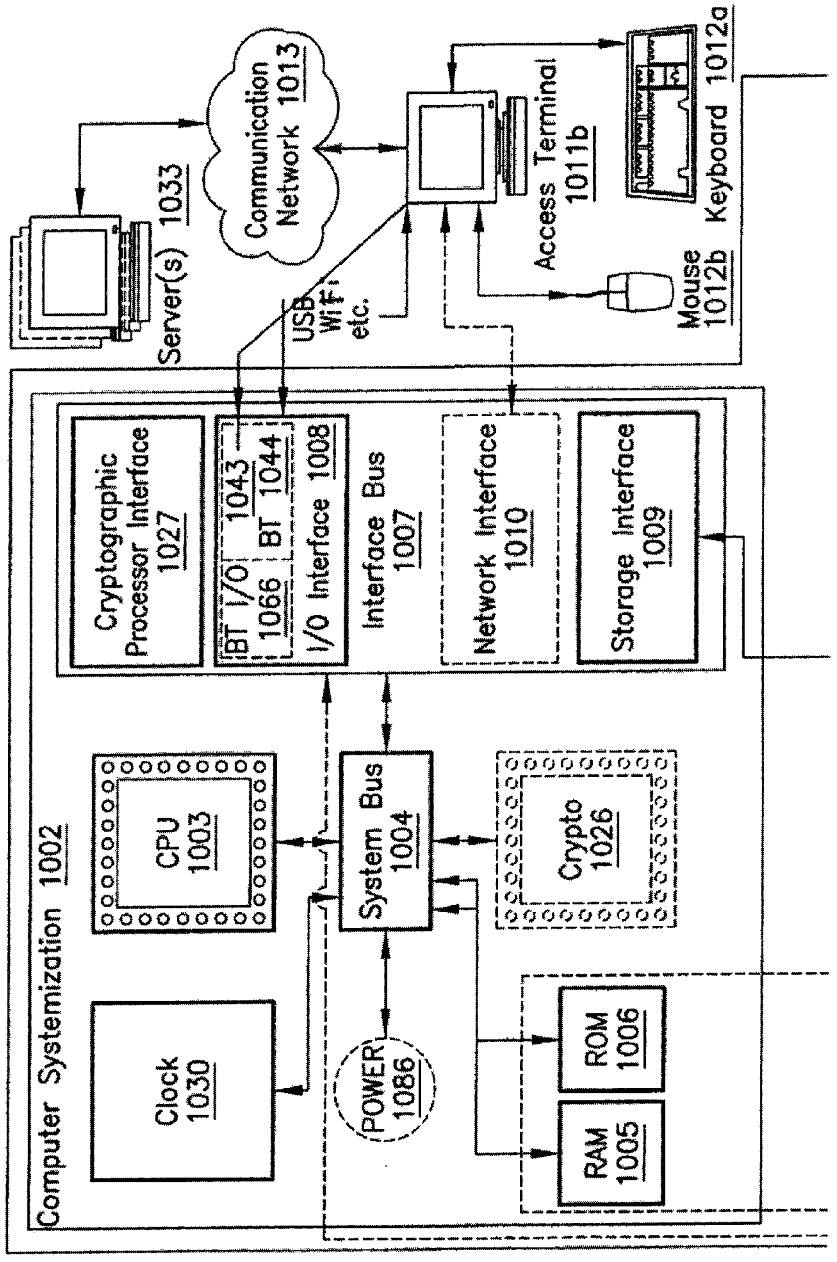

Server(s) 1033
Communication Network 1013
Access Terminal 1011b
Keyboard 1012a
Mouse 1012b
USB Wi Fi etc.
Computer Systemization 1002
Cryptographic Processor Interface 1027
BT I/O 1066
1043
BT 1044
I/O Interface 1008
Interface Bus 1007
Network Interface 1010
Storage Interface 1009
CPU 1003
System Bus 1004
Crypto 1026
Clock 1030
POWER 1086
ROM 1006
RAM 1005

Fig. 10a

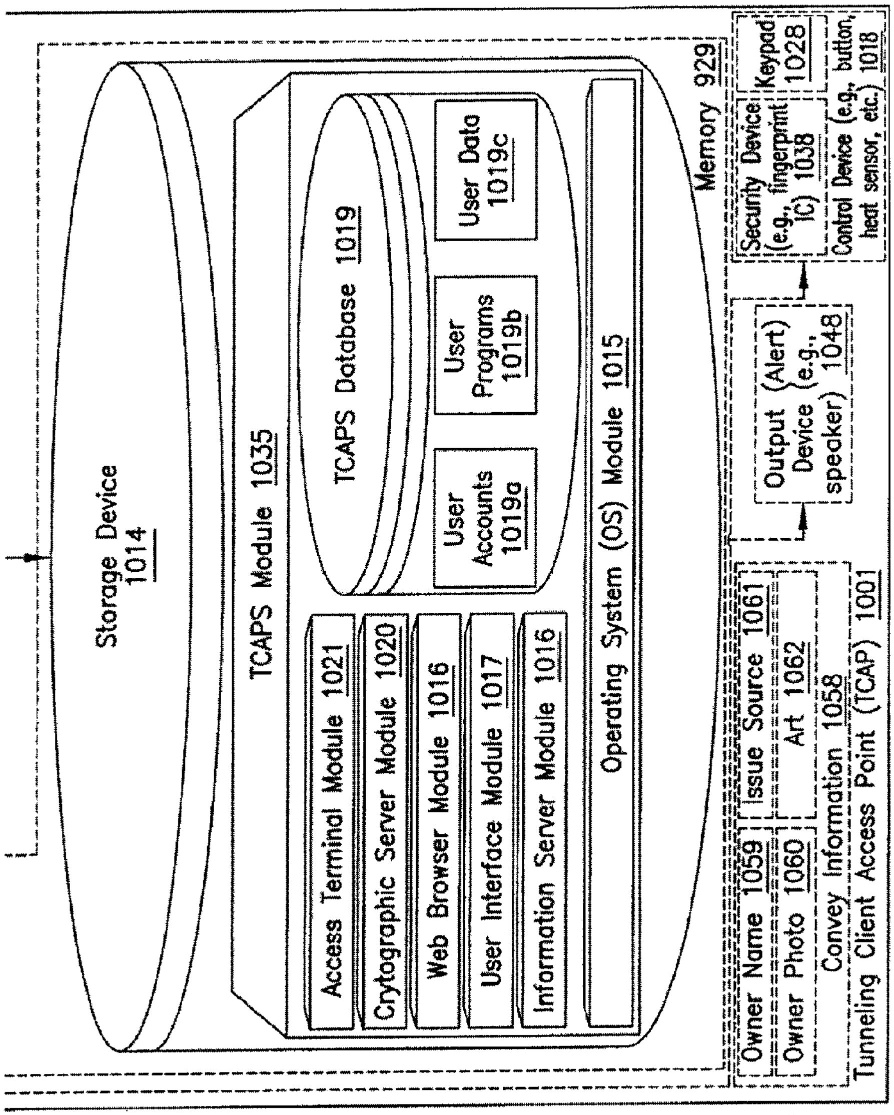

Storage Device 1014
TCAPS Module 1035
TCAPS Database 1019
User Accounts 1019a
User Programs 1019b
User Data 1019c
Access Terminal Module 1021
Crytographic Server Module 1020
Web Browser Module 1016
User Interface Module 1017
Information Server Module 1016
Operating System (OS) Module 1015
Memory 929
Owner Name 1059
Issue Source 1061
Owner Photo 1060
Art 1062
Convey Information 1058
Tunneling Client Access Point (TCAP) 1001
Output (Alert) Device (e.g., speaker) 1048
Security Device (e.g., fingerprint IC) 1038
Keypad 1028
Control Device (e.g., button, heat sensor, etc.) 1018

Fig.10b

APPARATUS, METHOD AND SYSTEM FOR A TUNNELING CLIENT ACCESS POINT

This application is a continuation application of U.S. application Ser. No. 12/950,321, filed Nov. 19, 2010, which is a continuation application of U.S. application Ser. No. 10/807,731, filed on Mar. 23, 2003, now U.S. Pat. No. 7,861,006.

FIELD

The present invention is directed generally to an apparatus, method, and system of accessing data, and more particularly, to an apparatus, method and system to execute and process data by tunneling access through a terminal.

BACKGROUND

Portable Computing and Storage

Computing devices have been becoming smaller over time. Currently, some of the smallest computing devices are in the form of personal digital assistants (PDAs). Such devices usually come with a touch screen, an input stylus and/or mini keyboard, and battery source. These devices, typically, have storage capacities around 64 MB. Examples of these devices include Palm's Palm Pilot.

Information Technology Systems

Typically, users, which may be people and/or other systems, engage information technology systems (e.g., commonly computers) to facilitate information processing. In turn, computers employ processors to process information; such processors are often referred to as central processing units (CPU). A common form of processor is referred to as a microprocessor. A computer operating system, which, typically, is software executed by CPU on a computer, enables and facilitates users to access and operate computer information technology and resources. Common resources employed in information technology systems include: input and output mechanisms through which data may pass into and out of a computer; memory storage into which data may be saved; and processors by which information may be processed. Often information technology systems are used to collect data for later retrieval, analysis, and manipulation, commonly, which is facilitated through database software. Information technology systems provide interfaces that allow users to access and operate various system components.

User Interface

The function of computer interfaces in some respects is similar to automobile operation interfaces. Automobile operation interface elements such as steering wheels, gearshifts, and speedometers facilitate the access, operation, and display of automobile resources, functionality, and status. Computer interaction interface elements such as check boxes, cursors, menus, scrollers, and windows (collectively and commonly referred to as widgets) similarly facilitate the access, operation, and display of data and computer hardware and operating system resources, functionality, and status. Operation interfaces are commonly called user interfaces. Graphical user interfaces (GUIs) such as the Apple Macintosh Operating System's Aqua, Microsoft's Windows XP, or Unix's X-Windows provide a baseline and means of accessing and displaying information, graphically, to users.

Networks

Networks are commonly thought to comprise of the interconnection and interoperation of clients, servers, and intermediary nodes in a graph topology. It should be noted that the term "server" as used herein refers generally to a computer, other device, software, or combination thereof that processes and responds to the requests of remote users across a communications network. Servers serve their information to requesting "clients." The term "client" as used herein refers generally to a computer, other device, software, or combination thereof that is capable of processing and making requests and obtaining and processing any responses from servers across a communications network. A computer, other device, software, or combination thereof that facilitates, processes information and requests, and/or furthers the passage of information from a source user to a destination user is commonly referred to as a "node." Networks are generally thought to facilitate the transfer of information from source points to destinations. A node specifically tasked with furthering the passage of information from a source to a destination is commonly called a "router." There are many forms of networks such as Local Area Networks (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc. For example, the Internet is generally accepted as being an interconnection of a multitude of networks whereby remote clients and servers may access and interoperate with one another.

SUMMARY

Although all of the aforementioned portable computing systems exist, no effective solution to securely access, execute, and process data is available in an extremely compact form. Currently, PDAs, which are considered among the smallest portable computing solution, are bulky, provide uncomfortably small user interfaces, and require too much power to maintain their data. Current PDA designs are complicated and cost a lot because they require great processing resources to provide custom user interfaces and operating systems. Further, current PDAs are generally limited in the amount of data they can store or access. No solution exists that allows users to employ traditional large user interfaces they are already comfortable with, provides greater portability, provides greater memory footprints, draws less power, and provides security for data on the device. As such, the disclosed tunneling client access point (TCAP) is very easy to use; at most it requires the user to simply plug the device into any existing and available desktop or laptop computer, through which, the TCAP can make use of a traditional user interface and input/output (I/O) peripherals, while the TCAP itself, otherwise, provides storage, execution, and/or processing resources. Thus, the TCAP requires no power source to maintain its data and allows for a highly portable "thumb" footprint. Also, by providing the equivalent of a plug-n-play virtual private network (VPN), the TCAP provides certain kinds of accessing of remote data in an easy and secure manner that was unavailable in the prior art.

In accordance with certain aspects of the disclosure, the above-identified problems of limited computing devices are overcome and a technical advance is achieved in the art of portable computing and data access. An exemplary tunneling client access point (TCAP) includes a method to dispose a portable storage device in communication with a terminal. The method includes providing the memory for access on the terminal, executing processing instructions from the memory on the terminal to access the terminal, communicating through a conduit, and processing the processing instructions.

In accordance with another embodiment, a portable tunneling storage processor is disclosed. The apparatus has a memory and a processor disposed in communication with the memory, and configured to issue a plurality of processing instructions stored in the memory. Also, the apparatus has a conduit for external communications disposed in communication with the processor, configured to issue a plurality of communication instructions as provided by the processor, configured to issue the communication instructions as signals to engage in communications with other devices having compatible conduits, and configured to receive signals issued from the compatible conduits.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate various non-limiting, example, inventive aspects in accordance with the present disclosure:

FIG. **1** is of a flow diagram illustrating embodiments of a tunneling client access point (TCAP);

FIG. **2** is of a flow diagram illustrating embodiments of a system of tunneling client access point and access terminal interaction;

FIG. **3** is of a flow diagram illustrating embodiments of engaging the tunneling client access point to an access terminal interaction;

FIG. **4** is of a flow diagram illustrating embodiments of accessing the tunneling client access point and server through an access terminal;

FIGS. **5-8** is of a flow diagram illustrating embodiments of facilities, programs, and/or services that the tunneling client access point and server may provide to the user as accessed through an access terminal;

FIG. **9** is of a block diagram illustrating embodiments of a tunneling client access point server controller;

FIG. **10** is of a block diagram illustrating embodiments of a tunneling client access point controller;

The leading number of each reference number within the drawings indicates the first figure in which that reference number is introduced. As such, reference number **101** is first introduced in FIG. **1**. Reference number **201** is first introduced in FIG. **2**, etc.

DETAILED DESCRIPTION

Topology

FIG. **1** illustrates embodiments for a topology between a tunneling client access point (TCAP) (see FIG. **10** for more details on the TCAP) and TCAP server (TCAPS) (see FIG. **9** for more details on the TCAPS). In this embodiment, a user **133***a* may plug-in a TCAP into any number of access terminals **127** located anywhere. Access terminals (ATs) may be any number of computing devices such as servers, workstations, desktop computers, laptops, portable digital assistants (PDAs), and/or the like. The type of AT used is not important other than the device should provide a compatible mechanism of engagement to the TCAP **130** and provide an operating environment for the user to engage the TCAP through the AT. In one embodiment, the TCAP provides a universal serial bus (USB) connector through which it may plug into an AT. In other embodiment, the TCAP may employ Bluetooth, WiFi and/or other wireless connectivity protocols to connect with ATs that are also so equipped. In one embodiment, the AT provides Java and/or Windows runtime environments, which allows the TCAP to interact with the input/output mechanisms of the AT. See FIG. **9** for more details and embodiments on the types of connections that may be employed by the TCAP. Once the TCAP has engaged with an AT, it can provide the user with access to its storage and processing facilities.

If the AT is connected to a communication network **113**, the TCAP may then communicate beyond the AT. In one embodiment, the TCAP can provide extended storage and/or processing resources by engaging servers **110**, **115**, **120**, which have access to and can provide extended storage **105** to the TCAP through the AT. In one embodiment, a single server and storage device may provide such TCAP server support. In another embodiment, server support is provided over a communications network, e.g., the Internet, by an array of front-end load-balancing servers **120**. These servers can provide access to storage facilities within the servers or to remote storage **105** across a communications network **113***b, c* (e.g., a local area network (LAN)). In such an embodiment, a back-end server **110** may offload the front-end server with regard to data access to provide greater throughput. For purposes of load balancing and/or redundancy, a backup server **115** may be similarly situated to provide for access and backup in an efficient manner. In such an embodiment, the back-end servers may be connected to the front-end servers through a communications network **113***b* (e.g., wide area network (WAN)). The backend servers **110**, **115** may be connected to the remote storage **105** through a communications network **113***c* as well (e.g., a high speed LAN, fiber-channel, and/or the like).

Thus, to the user **133***a*, the contents of the TCAP **130** appear on the AT as being contained on the TCAP **125** even though much of the contents may actually reside on the servers **115**, **120** and/or the servers' storage facilities **105**. In these ways, the TCAP "tunnels" data through an AT. The data may be provided through the AT's I/O for the user to observe without it actually residing on the AT. Also, the TCAP may tunnel data through an AT across a communications network to access remote servers without requiring its own more complicated set of peripherals and I/O.

TCAP and AT Interaction

FIG. **2** illustrates embodiments for a system of tunneling client access point (TCAP) (see FIG. **10** for more details on the TCAP) and access terminal interaction. FIG. **2** provides an overview for TCAP and AT interaction and subsequent figures will provide greater detail on elements of the interaction. In this embodiment, a user engages the TCAP **201**. For example, the user may plug the TCAP into an AT via the AT's USB port. Thereafter the user is presented with a login prompt **205** on the AT's display mechanism, e.g., on a video monitor. After a user successfully logs in (for example by providing a user name and password) **204**, the TCAP can then accept user inputs from the AT and its peripherals (the TCAP can then also provide output to the user via the AT's peripherals).

The user may employ the AT's input peripherals as user input devices that control actions on the TCAP. Depending on the user's actions **215**, the TCAP can be used by the AT as a storage device from which it can access and store data and programs **225**. For example, if the user takes the action of opening a file from the TCAP's memory, e.g., by double clicking on an icon when the TCAP is mounted as a USB drive on the AT, then the AT may treat the TCAP as a memory device and retrieve information from the TCAP **225**. If the user's action **215** is one that is directed at executing on the TCAP **215**, then the AT will not be involved in any execution. For example, if the user drops an icon representing a graphics file onto a drag-and-drop location visually representing the TCAP, then the file may be copied to the TCAP where it will process and spool the file for sending the graphics file to be printed at a remote location. In such a case, all of the requirements to process and spool the file are handled by the TCAP's processor and the AT would only be used as a mechanism for user input and output and as a conduit through which the TCAP may send files.

Regardless of if there is an action **215** to execute on the TCAP **220** or to access or store data on the TCAP **225**, the AT is used to display the status of any actions **230**. At any time the user may select to terminate TCAP related facilities executing either on the AT, a backend server, on the TCAP itself, and/or the like **235**. In one embodiment, the user may select a quit option that is displayed on the AT's screen. In another embodiment, the user may simply disengage the TCAP from the AT by severing the connection (e.g., turning power off, physically pulling the device off the AT, turning off wireless transmissions, and/or the like). It should be noted that such abrupt severing may result in the loss of data, file corruption, etc. if the TCAP has not saved data that is on the AT or on some remote server, however, if the TCAP is employing flash like memory, its contents should remain intact.

If there is no instruction signal to terminate the TCAP **235**, execution will continue and the TCAP will continue to take and look for input from the user. Of course if the TCAP has been set to perform certain actions, those actions will continue to execute, and the TCAP may respond to remote servers when it is communicating with them through the AT. When the user issues a terminate signal **235**, then the TCAP will shut down by saving any data to the TCAP that is in the AT's memory and then terminating any programs executing on both the AT and TCAP that were executed by and/or from the TCAP **240**. If no activities are taking place on the TCAP and all the data is written back to the TCAP **240**, then the TCAP may optionally unmount itself from the AT's file-system **245**. At this point, if there is a TCAP I/O driver executing on the AT, that driver may be terminated as trigered by the absence of the TCAP at a mount point **250**. After the TCAP is unmounted and/or the TCAP I/O driver is terminated, it is safe to disengage the TCAP from the AT.

TCAP and AT Interaction

FIG. **3** illustrates embodiments engaging the tunneling client access point to an access terminal interaction. Examples of engaging the TCAP **301** with an AT were discussed above in FIG. **1 127**, **130**, **133***a* and FIG. **2 201**. In one embodiment, the TCAP **130** is engaged with an access terminal **327**, **305**. As mentioned in FIG. **1**, the TCAP is capable of engaging with ATs using a number of mechanisms. In one embodiment, the TCAP has a USB connector for plugging into an AT, which acts as a conduit for power and data transfer. In another embodiment, the TCAP may use Bluetooth to establish a wireless connection with a number of ATs. In another embodiment, the TCAP may employ WiFi. In yet another embodiment, the TCAP may employ multiple communications mechanisms. It should be noted, with some wireless mechanisms like Bluetooth and WiFi, simply coming into proximity with an AT that is configured for such wireless communication may result in the TCAP engaging with and establish a communications link with the AT. In one embodiment, the TCAP has a "connect" button that will allow such otherwise automatically engaging interactions take place only if the "connect" button is engaged by a user. Such an implementation may provide greater security for users (see FIG. **10** for more details on the TCAP).

After being engaged **305**, the TCAP will then power on. In an embodiment requiring a direct connection, e.g., USB, simply plugging the TCAP into the AT provides power. In a wireless embodiment, the TCAP may be on in a lower powered state or otherwise turned on by engaging the connect button as discussed above. In such an embodiment, the TCAP can employ various on-board power sources (see FIG. **10** for more details on the TCAP). The TCAP then may load its own operating system **315**. The operating system can provide for interaction with the AT. In one embodiment, a Java runtime is executed on the TCAP, and Java applets communicate with the AT through Java APIs. In another embodiment, a driver is loaded onto the AT, and the on-TCAP Java operating system applets communicate to and through the AT via the driver running on the AT, wherein the driver provides an API through and to which messages may be sent.

After engaging with the AT, the TCAP can provide its memory space to the AT **320**. In one embodiment, the TCAP's memory is mapped and mounted as a virtual disk drive **125** storage **325**. In this manner, the TCAP may be accessed and manipulated as a standard storage device through the AT's operating system. Further, the TCAP and in some cases the AT can determine if the AT is capable of accessing program instructions stored in the TCAP's memory **330**. In one embodiment, the AT's operating system looks to auto-run a specified file from any drive as it mounts. In such an embodiment, the TCAP's primary interface may be specified in such a boot sequence. For example, under windows, an autorun.inf file can specify the opening of a program from the TCAP by the AT; e.g., OPEN=TCAP.EXE.

Many operating systems are capable of at least accessing the TCAP as a USB memory drive **330** and mounting its contents as a drive, which usually becomes accessible in file browsing window **125**. If the TCAP does not mount, the AT's operating system will usually generate an error informing the user of a mounting problem. If the AT is not capable of executing instruction from the TCAP, a determination is made if an appropriate driver is loaded on the AT to access the TCAP **335**. In one embodiment, the TCAP can check to see if an API is running on the AT. For example, the TCAP provide an executable to be launched, e.g., as specified through autorun.inf, and can establish communications through its connection to the AT, e.g., employing TCP/IP communications over the USB port. In such an embodiment, the TCAP can ping the AT for the program, and if an acknowledgement is received, the TCAP has determined that proper drivers and APIs exist. If no such API exists, the TCAP may launch a driver installation program for the AT as through an autorun.inf. In an alternative embodiment, if nothing happens, a user may double click onto an installer program that is stored on the mounted TCAP **342**, **340**. It should be noted, that although the TCAP's memory space may be mounted, certain areas of the TCAP may be inaccessible until there is an authorization. For example, certain areas and content on the TCAP may be encrypted. It should be noted that any such access terminal modules that drive AT and TCAP interaction may be saved onto the TCAP by copying the module to a mounted TCAP. Nevertheless, if the AT is capable of accessing program instructions in TCAP memory **330**, a TCAP driver is loaded on the AT **335**, and/or the user engages a program in the TCAP memory **340**, then the AT can execute program instructions from the TCAP's memory, which allows the TCAP to use the AT's I/O and allowing the user to interface with TCAP facilities **345**. It should be noted that some ATs may not be able to mount the TCAP at all. In such an instance, the user may have to install the TCAP drivers by downloading them from a server on the Internet, loading them from a diskette or CD, and/or the like. Once the TCAP is engaged to the AT **301**, execution may continue **398**.

TCAP and AT Interaction

FIG. **4** illustrates embodiments accessing the tunneling client access point and server through an access terminal. Upon engaging the TCAP to the AT as described in FIG. **3** **301**, **398**, the user may then go on to access the TCAP and its services **498**. It should be noted that users may access certain unprotected areas of the TCAP once it has been mounted, as described in FIG. **3**. However, to more fully access the TCAP's facilities, the user may be prompted to either login and/or registration window **205***a* to access the TCAP and its services, which may be displayed on the AT **405**. It is important to note that in one embodiment, the execution of the login and/or registration routines are handled by the TCAP's processor. In such an embodiment, the TCAP may run a small Web server providing login facilities, and connect to other Web based services through the AT's connection to the Internet. Further, the TCAP may employ a basic Web browsing core engine by which it may connect to Web services through the AT's connection to a communications network like the Internet. For purposes of security, in one embodiment, the TCAP may connect to a remote server by employing a secure connection, e.g., HTTPS, VPN, and/or the like.

Upon displaying a login window **405**, e.g., **205***a*, the user may select to register to access the TCAP and its services, or they may simply log in by providing security verification. In one example, security authorization may be granted by simply providing a user and password as provided through a registration process. In another embodiment, authorization may be granted through biometric data. For example, the TCAP may integrate a fingerprint and/or heat sensor IC into its housing. Employing such a device, and simply by providing one's finger print by laying your finger to the TCAP's surface, would provide the login facility with authorization if the user's finger print matches one that was stored during the registration process.

If the user does not attempt to login **415**, i.e., if the user wishes to register to use the TCAP and its services, then the TCAP can determine if the AT is online **420**. This may be accomplished in a number of ways. In one embodiment, the TCAP itself may simply ping a given server and if acknowledgement of receipt is received, the TCAP is online. In another embodiment, the TCAP can query for online status by engaging the AT through the installed APIs. If the AT is not online, then the user may be presented with an error message **425**. Thus, if a user does not have a login, and does not have the ability to register, then restricted areas of the TCAP will remain unavailable. Thereafter, flow can continue **498** and the user may have another opportunity to login and/or register. In one embodiment as a login integrity check, the TCAP keeps track of the number of failed attempts to login and/or register and may lock-out all further access if a specified number of failed attempts occurs. In one embodiment, the lockdown may be permanent by erasing all data on the TCAP. In another embodiment, the TCAP will disallow further attempts for a specified period of time.

If the user is attempting to register **415**, and the AT is online **420**, then the user map provide registration information **440** into a screen form **440***a*. Registration information fields may require a user's name, address, email address, credit card information, biometric information (e.g., requiring the user to touch a biometric fingerprint IC on the TCAP), and/or the like. The TCAP may determine if all the information was provided as required for registration and may query backend servers to determine if the user information is unique **445**. If the user did not properly fill out the registration information or if another user is already registered, the TCAP can provided an error message to such effect. Also, both the TCAP and its back-end servers may make log entries tracking such failed attempts for purposes of defending against fraud and/or security breaches. The user may then modify the registration information **440** and again attempt to register. Similarly to the login integrity checks, the TCAP can lockout registration attempts if the user fails to register more than some specified number of times.

Upon providing proper registration information **445** or proper login authentication **415**, the TCAP can query backend servers to see if the user is registered. In one embodiment, such verification may be achieved by sending a query to the

servers to check its database for the authorization information and/or for duplicate registrations. The servers would then respond providing an acknowledgment of proper registration and authorization to access data on the backend servers. If the users are not registered on the backend servers **430**, then the TCAP can provide an error message to the user for display on the AT to such effect **435**. In an alternative embodiment, the registration information may be stored on the TCAP itself. In one embodiment, the registration would be maintained in encrypted form. Thus, the user's login information may be checked relative to the information the TCAP itself, and if there is a match, access may be granted, otherwise an error message will be displayed **435**. The TCAP may then continue **498** to operate as if it were just engaged to the AT.

If the user is confirmed to be registered **430**, then the TCAP may provide options for display **453**, **453***a*. Depending on the context and purpose of a particular TCAP, the options may vary. For example, the a screen **453***a* may provide the user with the options to access data either online or offline. The user might simply click on a button and gain secure access to such data that may be decrypted by the TCAP. In one embodiment, the TCAP will determine if the AT is online **455**. If this was already determined **420**, this check **455** may be skipped.

If the AT is online **455**, optionally, the TCAP determines if the user wishes to synchronize the contents of the TCAP with storage facilities at the backend server **470**. In one embodiment, the user may designate that such synchronization is to always take place. If synchronization is specified **470**, then the TCAP will provide and receive updated data to and from the backend servers, overwriting older data with updated versions of the data **475**. If the AT is online **455** and/or after any synchronization **475**, the TCAP may provide the user with all of its service options as authorized by the account and programs available on the TCAP and at the backend server **480**. Once again, these facilities, programs, and/or services may vary greatly depending on the context and deployment requirements of the user. The options to be presented to the user from the TCAP or the TCAP services from the backend server, as displayed through the TCAP onto the AT's display **480**, are myriad and some example embodiments are provided in FIGS. **5**-**8**. Upon presenting the user with the options, the user is then able to access, execute, store data and programs on the TCAP and on the remote server **485**. All areas of the TCAP and services are then open, including any encrypted data areas.

If the AT is not online **455**, the TCAP may provide options for the user not including online services **460**. In one embodiment, the online options that may be presented on the AT display will be dimmed and/or omitted to reflect the lack of accessibility. However, the user will be able to access, execute, store data and programs on the TCAP, including any encrypted data areas **465**.

TCAP Facilities and Services

FIGS. **5**-**8** illustrate embodiments of facilities, programs, and/or services that the tunneling client access point and server may provide to the user as accessed through an AT. Any particular set of facilities may have a myriad of options. The options and the general nature of the facilities provided on any particular TCAP are dependant upon the requirements of a given set of users. For example, certain groups and/or agencies may require TCAPS to be targeted towards consumer photographs, and may employ TCAPs to further that end. Other groups may require high security facilities, and tailor the TCAPs accordingly. In various environments, an organization may wish to provide a secure infrastructure to all of its agents for securely accessing the organization's data from anywhere and such an organization could tailor the TCAPs contents to reflect and respond to its needs. By providing a generalized infrastructure on the TCAP backend servers and within the TCAP by using a generalized processor, the TCAPs may be deployed in numerous environments.

In one particular embodiment as in FIG. **5**, the TCAP provides facilities to access, process, and store email, files, music, photos and videos through the TCAP. Upon engaging **101** of FIG. **1** the TCAP **130** to an AT **307**, the TCAP will mount and display through the AT's file browser window **125** of FIG. **1**. As has already described, in the case where the AT has no TCAP driver software, the user may double click on the installer software stored on the TCAP **507**. Doing so will launch the installer software from the TCAP's memory to execute on the AT, and the user may be presented with a window to confirm the desire to install the TCAP software onto the AT **507**. Upon confirming the install **507**, the software will install on the AT and the user will be asked to wait as they are apprised of the install progress **509**.

Upon installation, the TCAP front-end software may execute and present the user with various options in various and fanciful interface formats **511**, **460**, **480** of FIG. **4**. In one embodiment, these user interfaces and programs are Java applications that may execute on the AT and a present Java runtime. In an alternative embodiment, a small applet may run on the AT, but all other activities may execute on the TCAP's processor, which would use the AT display only as a display terminal. In the embodiment where the TCAP executes program instructions, the TCAP may be engaged to receive commands and execute by receiving a signal from the access terminal driver instructing it to execute certain program files or, alternatively, looking to default location and executing program instructions. In yet another embodiment, the TCAP may obtain updated interfaces and programs from a backend server for execution either on the TCAP itself and/or the AT; this may be done by synchronization with the backend server and checking for updates of specified files at the backend server. By engaging the user interface, perhaps by clicking on a button to open the TCAP facilities and services **511**, the interface may further unfurl to present options to access said facilities and services **513**. Here, the interface may reflect ownership of the TCAP by providing a welcome screen and showing some resources available to the user; for example, a button entitled "My Stuff" may serve as a mechanism to advance the user to a screen where they may access their personal data store. At this point the user may attempt to login to access their data by engaging an appropriate button, which will take them to a screen that will accept login information **519**. Alternatively, the user may also register if it is their first time using the TCAP by selecting an appropriate button, which will advance the user to a registration screen **515** wherein the user may enter their name, address, credit card information, etc. Upon successfully providing registration information, the user may be prompted for response to further solicitations on a follow-up screen **517**. For example, depending on the services offered for a particular TCAP, the user may be provided certain perks like **5** MB of free online storage on a backend server, free photographic prints, free email access, and/or the like **517**.

After the user is prompted to login **518** and successfully provides proper login information **519**, or after successfully registering **515** and having responded to any solicitations **517**, the user may be provided with general options **521** to access data stored on the TCAP itself **522** or in their online account **520** maintained on a backend server. For example, if the user selects the option to access their online storage **520**, they may be presented with more options to interact with email, files, music, photos and videos that are available online

523. Perhaps if the user wished to check their email, the user might select to interact with their email, and a screen allowing them to navigate through their email account(s) would be presented **525**. Such online access to data may be facilitated through http protocols whereby the TCAP applications send and receive data through http commands across a communications network interacting with the backend servers and/or other servers. Any received results may be parsed and imbedded in a GUI representation of a Java application. For example, the email facility may run as a Java applet **525** and may employ a POP mail protocol to pull data from a specified mail server to present to the user.

Similarly, many other facilities may be engaged by the user through the TCAP. In one embodiment, the user may drag **508** a file **506** onto a drag-and-drop zone **505** that is presented on the TCAP interface. Upon so doing, various drag-and-drop options may unfurl and present themselves to the user **550**. It should be noted that the file may come from anywhere, i.e., from the AT, the TCAP, and/or otherwise. For example, upon dragging and dropping a graphics file, a user may be prompted with options to order prints, upload the file to an online storage space, save the file to the TCAP's memory space, cancel the action, and/or the like **550**. If the user sends the file for storage, or otherwise wishes to see and manage their data, an interface allowing for such management may be presented **555**. The interface may organize and allow access to general data, picture, and music formats **554**, provide usage statistics (e.g., free space, capacity, used space, etc.) **553**, provide actions to manipulate and organize the data **552**, provide status on storage usage on the TCAP **551** and online **549**, and/or the like.

Should the user engage a user interface element indicating the wish to manipulate their picture data **548**, the TCAP interface will update to allow more specific interaction with the user's photos **557**. In such a screen, the user may select various stored pictures and then indicate a desire to order photo prints by engaging the appropriate user interface element **558**. Should the user indicate their desire for prints **558**, they will be presented with an updated interface allowing the specification of what graphics files they wish to have printed **559**. In one embodiment, the users may drag-and-drop files into a drop zone, or otherwise engage file browsing mechanisms **560** that allow for the selection of desired files. Upon having identified the files for prints **559**, a user may be presented with an interface allowing for the selection of print sizes and quantities **561**. After making such specifications, the user may be required to provide shipping information **563** and information for payments **565**. After providing the billing information to a backend server for processing and approval, the user may be presented with a confirmation interface allowing for editing of the order, providing confirmation of costs, and allowing for submission of a final order for the selected prints **567**. Upon submitting the order, the TCAP will process the files for spooling to a backend server that will accept the order and files, which will be developed as prints and the user's account will be charged accordingly. In one embodiment, all of the above order and image processing operations occur and execute on the TCAP CPU. For example, the TCAP may employ various rendering technologies, e.g., ghostscript, to allow it to read and save PDFs and other media formats.

FIG. **6** goes on to illustrate embodiments and facets of the facilities of FIG. **5**. The TCAP interface allows the user to perform various actions at any given moment. As has already been discussed in FIG. **5**, the user may drag **508** a file **506** onto a drag and drop zone **505** so as to provide the file to the TCAP for further manipulation. As in **550** of FIG. **5**, the user may be presented with various options subsequent to a drag-and-drop operation. Also, the TCAP interface may provide visual feedback that files have been dropped in the drop zone by highlighting the drop zone **505***b*. Should the user wish, they may close the TCAP interface by engaging a close option **633**. Also, the ability to change and/or update their personal information may be accessed through the TCAP interface **616**, which would provide a form allowing the user to update their registration information **630**. In one embodiment, should the user forget their login information, they may request login help **635** and the TCAP will send their authorization information to the last known email address and inform the user of same **640**. Also, the TCAP interface may provide help facilities that may be accessed at any time by simply engaging a help facility user interface element **617**. So doing will provide the user with help screen information as to how to interact with the TCAP's facilities **625**.

Upon providing proper login information **619** and logging-in **619**, the user may be presented with a welcome screen with various options to access their data **621** as has already been discussed in FIG. **5**, **521**. By engaging a user interface element to access online storage **620**, the user may be presented with various options to interact with online storage **623**, **523** of FIG. **5**. Should the user wish to interact with data on the TCAP itself, the user may indicate so by engaging the appropriate user interface option **622**. So doing will provide the user with further options related to data stored on the TCAP **655**. The user may engage an option to view the storage contents **658** and the TCAP interface will provide a listing of the contents **662**, which may be manipulated through selection and drag-and-drop operations with the files.

In one embodiment, the user may order prints of photos **657** from files that are on the TCAP itself. As discussed in FIG. **5**, the user may select files for which they desire prints **660**. Here, the selected files will first be processed by the TCAP in preparation for sending to backend servers and file manipulations **670**. The user may specify various attributes regarding the prints they desire, e.g., the size, number, cropping, red-eye correction, visual effects, and/or the like **661**. In one embodiment, such processing occurs on the TCAP processor, while in other embodiments such processing can take place on the AT or backend server. Once again, the user may provide a shipping address **663**, and make a final review to place the order **667**. Upon committing to the order **667**, the processed files are uploaded to the backend servers that will use the files to generate prints **690**. A confirmation screen may then be provided to the user with an order number and other relevant information **695**.

FIG. **7** goes on to illustrate embodiments and facets of the facilities of FIGS. **5**-**6** as may apply in different environments. As is demonstrated, the look and feel of the TCAP interface is highly malleable and can serve in many environments. FIG. **7** illustrates that even within a single organization, various environments might benefit from TCAPs and services tailored to serve such environments **733***b*-*d*. In this case TCAPs can serve in consumer **733***b*, industry trade **733***c*, corporate **733***d*, and/or the like environments.

As has already been discussed, initially in any of the environments, after engaging the TCAP to an AT, the user may be prompted to install the TCAP interface **705** and informed of the installation procedure **710**. The user may then be presented with the installed TCAP interface **715**, which may be activated by engaging an interface element to unfurl the interface, e.g., in this case by opening the top to a can of soda **717**. Opening the interface will present the user with various options as **720**, as has already been discussed in FIGS. **5**-**6**. Similarly the user may login **725** or make a selection to

register for various TCAP services and provide the requisite information in the provided form **730**. Upon registering and/or logging-in **725**, various options may be presented based upon the configuration of the TCAP. For example, if the TCAP was configured and tailored for consumers, then upon logging in **725** the consumer user might be presented **733***a*-*b* with various consumer related options **740**. Similarly, if the TCAP were tailored for **733***a, c* the trade industry or **733***a, d* the corporate environment, options specific to the trade industry **770** and corporate environment **760** may be presented.

In one embodiment, an organization wishing to provide TCAPs to consumers might provide options **740** for free music downloads **743**, free Internet radio streaming **748**, free news (e.g., provided through an RSS feed from a server) **766**, free photo printing **750**, free email **740**, free coupons **742**, free online storage **741**, and/or the like. Users could further engage such services (e.g., clicking free music file links for downloading to the TCAP, by ordering prints **750**, etc. For example, the user may select files on the TCAP **750**, select the types of photos they would like to receive **752**, specify a delivery address **754**, confirm the order **756** all of which will result in the TCAP processing the files and uploading them to the backend servers for generation of prints (as has already been discussed in FIGS. **5**-**6**).

In another embodiment, an organization wishing to provide TCAPs to a trade industry might provide options **770** for advertising **780**, events **775**, promotions **772**, and/or the like. It is important to note that information regarding such options may be stored either on the TCAP or at a backend server. In one embodiment, such information may be constantly synchronized from the backend servers to the TCAPs. This would allow an organization to provide updates to the trade industry to all authorized TCAP "key holders." In such an embodiment, the user may be presented with various advertising related materials for the organization, e.g., print, television, outdoor, radio, web, and/or the like **780**. With regard to events, the user may be presented with various related materials for the organization, e.g., trade shows, music regional, sponsorship, Web, and/or the like **775**. With regard to promotions, the user may be presented with various related materials for the organization, e.g., rebates, coupons, premiums, and/or the like **772**.

In another embodiment, an organization wishing to provide TCAPs to those in the corporate environment and might provide options relating to various corporate entities **760**. Selecting any of the corporate entities **760** may provide the user with options to view various reports, presentations, and/or the like, e.g., annual reports, 10K reports, and/or the like **765**. Similarly, the reports may reside on the TCAP and/or the corporate TCAP can act as a security key allowing the user to see the latest corporate related materials from a remote backend server.

FIG. **8** goes on to illustrate embodiments and facets of the facilities of FIGS. **5**-**7** as may apply in different environments. FIG. **8** illustrates that TCAPs may serve to provide heightened security to any environment. As has been discussed in previous figures, users may engage the TCAP interface **805** to access various options **810**. The TCAP interface is highly adaptable and various services may be presented within it. For example, a stock ticker may be provided as part of the interface in a financial setting **810**. Any number of live data feeds may dynamically update on the face of the interface. Upon logging-in **815** or registering a new account **820**, the user may be informed that communications that are taking place are secured **825**. In one embodiment, various encryption formats may be used by the TCAP to send information securely to the backend servers. It is important to note that in such an embodiment, even if data moving out of the TCAP and across the AT were captured at the AT, such data would not be readable because the data was encrypted by the TCAP's processor. As such, the TCAP acts as a "key" and provides a plug-and-play VPN to users. Such functionality, heretofore, has been very difficult to set up and/or maintain. In this way, all communications, options presented and views of user data are made available only to the TCAP with the proper decryption key. In heightened security environments, display of TCAP data is provided on the screen only in bitmapped format straight to the video memory of the AT and, therefore, is not stored anywhere else on the AT. This decreases the likelihood of capturing sensitive data. As such, the user may access their data on the TCAP and/or online **830** in a secure form whereby the user may navigate and interact with his/her data and various services **835** in a secure manner.

Tunneling Client Access Point Server Controller

FIG. **9** illustrates one embodiment incorporated into a tunneling client access point server (TCAPS) controller **901**. In this embodiment, the TCAP controller **901** may serve to process, store, search, serve, identify, instruct, generate, match, and/or update data in conjunction with a TCAP (see FIG. **10** for more details on the TCAP). TCAPS act as back-end servers to TCAPs, wherein TCAPS provide storage and/or processing resources to great and/or complex for the TCAP to service itself. In effect, the TCAPS transparently extend the capacity of a TCAP.

In one embodiment, the TCAPS controller **901** may be connected to and/or communicate with entities such as, but not limited to: one or more users from user input devices **911**; peripheral devices **912**; and/or a communications network **913**. The TCAPS controller may even be connected to and/or communicate with a cryptographic processor device **928**.

A TCAPS controller **901** may be based on common computer systems that may comprise, but are not limited to, components such as: a computer systemization **902** connected to memory **929**.

Computer Systemization

A computer systemization **902** may comprise a clock **930**, central processing unit (CPU) **903**, a read only memory (ROM) **906**, a random access memory (RAM) **905**, and/or an interface bus **907**, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus **904**. Optionally, a cryptographic processor **926** may be connected to the system bus. The system clock typically has a crystal oscillator and provides a base signal. The clock is typically coupled to the system bus and various clock multipliers that will increase or decrease the base operating frequency for other components interconnected in the computer systemization. The clock and various components in a computer systemization drive signals embodying information throughout the system. Such transmission and reception of signals embodying information throughout a computer systemization may be commonly referred to as communications. These communicative signals may further be transmitted, received, and the cause of return and/or reply signal communications beyond the instant computer systemization to: communications networks, input devices, other computer systemizations, peripheral devices, and/or the like. Of course, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

The CPU comprises at least one high-speed data processor adequate to execute program modules for executing user and/or system-generated requests. The CPU may be a microprocessor such as AMD's Athlon, Duron and/or Opteron; IBM and/or Motorola's PowerPC; Intel's Celeron, Itanium, Pentium and/or Xeon; and/or the like processor(s). The CPU interacts with memory through signal passing through conductive conduits to execute stored program code according to conventional data processing techniques. Such signal passing facilitates communication within the TCAPS controller and beyond through various interfaces. Should processing requirements dictate a greater amount speed, mainframe and super computer architectures may similarly be employed.

Interface Adapters

Interface bus(ses) **907** may accept, connect, and/or communicate to a number of interface adapters, conventionally although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) **908**, storage interfaces **909**, network interfaces **910**, and/or the like. Optionally, cryptographic processor interfaces **927** similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. Interface adapters conventionally connect to the interface bus via a slot architecture. Conventional slot architectures may be employed, such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended) Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI(X)), Personal Computer Memory Card International Association (PCMCIA), and/or the like.

Storage interfaces **909** may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices **914**, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE), Institute of Electrical and Electronics Engineers (IEEE) 1394, fiber channel, Small Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

Network interfaces **910** may accept, communicate, and/or connect to a communications network **913**. Network interfaces may employ connection protocols such as, but not limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T, and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the like. A communications network may be any one and/or the combination of the following: a direct interconnection; the Internet; a Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network (WAN); a wireless network (e.g., employing protocols such as, but not limited to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A network interface may be regarded as a specialized form of an input output interface. Further, multiple network interfaces **910** may be used to engage with various communications network types **913**. For example, multiple network interfaces may be employed to allow for the communication over broadcast, multicast, and/or unicast networks. Input Output interfaces (I/O) **908** may accept, communicate, and/or connect to user input devices **911**, peripheral devices **912**, cryptographic processor devices **928**, and/or the like. I/O may employ connection protocols such as, but not limited to: Apple Desktop Bus (ADB); Apple Desktop Connector (ADC); audio: analog, digital, monaural, RCA, stereo, and/or the like; IEEE 1394a-b; infrared; joystick; keyboard; midi; optical; PC AT; PS/2; parallel; radio; serial; USB; video interface: BNC, composite, digital, Digital Visual Interface (DVI), RCA, S-Video, VGA, and/or the like; wireless; and/or the like. A common output device is a video display, which typically comprises a Cathode Ray Tube (CRT) or Liquid Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry and cable) that accepts signals from a video interface. The video interface composites information generated by a computer systemization and generates video signals based on the composited information in a video memory frame. Typically, the video interface provides the composited video information through a video connection interface that accepts a video display interface (e.g., a DVI connector accepting a DVI display cable).

User input devices **911** may be card readers, dongles, finger print readers, gloves, graphics tablets, joysticks, keyboards, mouse (mice), trackballs, trackpads, retina readers, and/or the like.

Peripheral devices **912** may be connected and/or communicate to I/O and/or other facilities of the like such as network interfaces, storage interfaces, and/or the like. Peripheral devices may be audio devices, cameras, dongles (e.g., for copy protection, ensuring secure transactions with a digital signature, and/or the like), external processors (for added functionality), goggles, microphones, monitors, network interfaces, printers, scanners, storage devices, video devices, visors, and/or the like.

It should be noted that although user input devices and peripheral devices may be employed, the TCAPS controller may be embodied as an embedded, dedicated, and/or headless device, wherein access would be provided over a network interface connection.

Cryptographic units such as, but not limited to, microcontrollers, processors **926**, interfaces **927**, and/or devices **928** may be attached, and/or communicate with the TCAPS controller. A MC68HC16 microcontroller, commonly manufactured by Motorola Inc., may be used for and/or within cryptographic units. Equivalent microcontrollers and/or processors may also be used. The MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the 16 MHz configuration and requires less than one second to perform a 512-bit RSA private key operation. Cryptographic units support the authentication of communications from interacting agents, as well as allowing for anonymous transactions. Cryptographic units may also be configured as part of CPU. Other commercially available specialized cryptographic processors include VLSI Technology's 33 MHz 6868 or Semaphore Communications' 40 MHz Roadrunner 184.

Memory

Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory **929**. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be understood that a TCAPS controller and/or a computer systemization may employ various forms of memory **929**. For example, a computer systemization may be configured wherein the functionality of on-chip CPU memory (e.g., registers), RAM, ROM, and any other storage devices are provided by a paper punch tape or paper punch card mechanism; of course such an embodiment would result in an extremely slow rate of operation. In a typical configuration, memory **929** will include ROM **906**, RAM **905**, and a storage device **914**. A storage device **914** may be any conventional computer system storage. Storage devices may include a drum; a (fixed and/or removable) magnetic disk drive; a magneto-optical drive; an optical drive (i.e., CD ROM/RAM/Recordable (R), ReWritable (RW), DVD R/RW, etc.); and/or other devices of the like. Thus, a computer systemization generally requires and makes use of memory.

Module Collection

The memory **929** may contain a collection of program and/or database modules and/or data such as, but not limited to: operating system module(s) **915** (operating system); information server module(s) **916** (information server); user interface module(s) **917** (user interface); Web browser module(s) **918** (Web browser); database(s) **919**; cryptographic server module(s) **920** (cryptographic server); TCAPS module(s) **935**; and/or the like (i.e., collectively a module collection). These modules may be stored and accessed from the storage devices and/or from storage devices accessible through an interface bus. Although non-conventional software modules such as those in the module collection, typically, are stored in a local storage device **914**, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through a communications network, ROM, various forms of memory, and/or the like.

Operating System

The operating system module **915** is executable program code facilitating the operation of a TCAPS controller. Typically, the operating system facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as Apple Macintosh OS X (Server), AT&T Plan 9, Be OS, Linux, Unix, and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Apple Macintosh OS, Microsoft DOS, Palm OS, Windows 2000/2003/3.1/95/98/CE/Millenium/NT/XP (Server), and/or the like. An operating system may communicate to and/or with other modules in a module collection, including itself, and/or the like. Most frequently, the operating system communicates with other program modules, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the interaction with communications networks, data, I/O, peripheral devices, program modules, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the TCAPS controller to communicate with other entities through a communications network **913**. Various communication protocols may be used by the TCAPS controller as a subcarrier transport mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the like.

Information Server

An information server module **916** is stored program code that is executed by the CPU. The information server may be a conventional Internet information server such as, but not limited to Apache Software Foundation's Apache, Microsoft's Internet Information Server, and/or the. The information server may allow for the execution of program modules through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-) C (++), Common Gateway Interface (CGI) scripts, Java, JavaScript, Practical Extraction Report Language (PERL), Python, WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program modules. After a Domain Name System (DNS) resolution portion of an HTTP request is resolved to a particular information server, the information server resolves requests for information at specified locations on a TCAPS controller based on the remainder of the HTTP request. For example, a request such as http://123.124.125.126/myInformation.html might have the IP portion of the request "123.124.125.126" resolved by a DNS server to an information server at that IP address; that information server might in turn further parse the http request for the "/myInformation.html" portion of the request and resolve it to a location in memory containing the information "myInformation.html." Additionally, other information serving protocols may be employed across various ports, e.g., FTP communications across port **21**, and/or the like. An information server may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the TCAPS database **919**, operating systems, other program modules, user interfaces, Web browsers, and/or the like.

Access to TCAPS database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the TCAP. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the TCAPS as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

Also, an information server may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

User Interface

A user interface module **917** is stored program code that is executed by the CPU. The user interface may be a conventional graphic user interface as provided by, with, and/or atop operating systems and/or operating environments such as Apple Macintosh OS, e.g., Aqua, Microsoft Windows (NT/XP), Unix X Windows (KDE, Gnome, and/or the like), and/or the like. The user interface may allow for the display, execution, interaction, manipulation, and/or operation of program modules and/or system facilities through textual and/or graphical facilities. The user interface provides a facility through which users may affect, interact, and/or operate a computer system. A user interface may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the user interface communicates with operating systems, other program modules, and/or the like. The user interface may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Web Browser

A Web browser module **918** is stored program code that is executed by the CPU. The Web browser may be a conventional hypertext viewing application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web browsing may be supplied with 128 bit (or greater) encryption by way of HTTPS, SSL, and/or the like. Some Web browsers allow for the execution of program modules through facilities such as Java, JavaScript, ActiveX, and/or the like. Web browsers and like information access tools may be integrated into PDAs, cellular telephones, and/or other mobile devices. A Web browser may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the Web browser communicates with information servers, operating systems, integrated program modules (e.g., plug-ins), and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. Of course, in place of a Web browser and information server, a combined application may be developed to perform similar functions of both. The combined application would similarly affect the obtaining and the provision of information to users, user agents, and/or the like from TCAPS enabled nodes. The combined application may be nugatory on systems employing standard Web browsers.

TCAPS Database

A TCAPS database module **919** may be embodied in a database and its stored data. The database is stored program code, which is executed by the CPU; the stored program code portion configuring the CPU to process the stored data. The database may be a conventional, fault tolerant, relational, scalable, secure database such as Oracle or Sybase. Relational databases are an extension of a flat file. Relational databases consist of a series of related tables. The tables are interconnected via a key field. Use of the key field allows the combination of the tables by indexing against the key field; i.e., the key fields act as dimensional pivot points for combining information from various tables. Relationships generally identify links maintained between tables by matching primary keys. Primary keys represent fields that uniquely identify the rows of a table in a relational database. More precisely, they uniquely identify rows of a table on the "one" side of a one-to-many relationship.

Alternatively, the TCAPS database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In another alternative, an object-oriented database may be used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can include a number of object collections that are grouped and/or linked together by common attributes; they may be related to other object collections by some common attributes. Object-oriented databases perform similarly to relational databases with the exception that objects are not just pieces of data but may have other types of functionality encapsulated within a given object. If the TCAPS database is implemented as a data-structure, the use of the TCAPS database may be integrated into another module such as the TCAPS module. Also, the database may be implemented as a mix of data structures, objects, and relational structures. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated. In one embodiment, the database module **919** includes three tables **919***a-c*. A user accounts table **919***a* includes fields such as, but not limited to: a user name, user address, user authorization information (e.g., user name, password, biometric data, etc.), user credit card, organization, organization account, TCAP unique identifier, account creation data, account expiration date; and/or the like. In one embodiment, user accounts may be activated only for set amounts of time and will then expire once a specified date has been reached. An user data table **919***b* includes fields such as, but not limited to: a TCAP unique identifier, backup image, data store, organization account, and/or the like. A user programs table **919***c* includes fields such as, but not limited to: system programs, organization programs, programs to be synchronized, and/or the like. In one embodiment, user programs may contain various user interface primitives, which may serve to update TCAPs. Also, various accounts may require custom database tables depending upon the environments and the types of TCAPs a TCAPS may need to serve. It should be noted that any unique fields may be designated as a key field throughout. In an alternative embodiment, these tables have been decentralized into their own databases and their respective database controllers (i.e., individual database controllers for each of the above tables). Employing standard data processing techniques, one may further distribute the databases over several computer systemizations and/or storage devices. Similarly, configurations of the decentralized database controllers may be varied by consolidating and/or distributing the various database modules **919***a-c*. The TCAPS may be configured to keep track of various settings, inputs, and parameters via database controllers.

A TCAPS database may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAPS database communicates with a TCAPS module, other program modules, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.

Cryptographic Server

A cryptographic server module **920** is stored program code that is executed by the CPU **903**, cryptographic processor **926**, cryptographic processor interface **927**, cryptographic processor device **928**, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic module; however, the cryptographic module, alternatively, may run on a conventional CPU. The cryptographic module allows for the encryption and/or decryption of provided data. The cryptographic module allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic module may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic module will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like. Employing such encryption security protocols, the TCAPS may encrypt all incoming and/or outgoing communications and may serve as node within a virtual private network (VPN) with a wider communications network. The cryptographic module facilitates the process of “security authorization” whereby access to a resource is inhibited by a security protocol wherein the cryptographic module effects authorized access to the secured resource. In addition, the cryptographic module may provide unique identifiers of content, e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A cryptographic module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. The cryptographic module supports encryption schemes allowing for the secure transmission of information across a communications network to enable a TCAPS module to engage in secure transactions if so desired. The cryptographic module facilitates the secure accessing of resources on TCAPS and facilitates the access of secured resources on remote systems; i.e., it may act as a client and/or server of secured resources. Most frequently, the cryptographic module communicates with information servers, operating systems, other program modules, and/or the like. The cryptographic module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

TCAPS

A TCAPS module **935** is stored program code that is executed by the CPU. The TCAPS affects accessing, obtaining and the provision of information, services, transactions, and/or the like across various communications networks. The TCAPS enables TCAP users to simply access data and/or services across a communications network in a secure manner. The TCAPS extends the storage and processing capacities and capabilities of TCAPs. The TCAPS coordinates with the TCAPS database to identify interassociated items in the generation of entries regarding any related information. A TCAPS module enabling access of information between nodes may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective−) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the TCAPS server employs a cryptographic server to encrypt and decrypt communications. A TCAPS module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAPS module communicates with a TCAPS database, operating systems, other program modules, and/or the like. The TCAPS may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Distributed TCAP

The structure and/or operation of any of the TCAPS node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the module collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

The module collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program modules in the program module collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program module instances and controllers working in concert may do so through standard data processing communication techniques.

The configuration of the TCAPS controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program modules, results in a more distributed series of program modules, and/or results in some combination between a consolidated and distributed configuration, data may be communicated, obtained, and/or provided. Instances of modules consolidated into a common code base from the program module collection may communicate, obtain, and/or provide data. This may be accomplished through intra-application data processing communication techniques such as, but not limited to: data referencing (e.g., pointers), internal messaging, object instance variable communication, shared memory space, variable passing, and/or the like.

If module collection components are discrete, separate, and/or external to one another, then communicating, obtaining, and/or providing data with and/or to other module components may be accomplished through inter-application data processing communication techniques such as, but not limited to: Application Program Interfaces (API) information passage; (distributed) Component Object Model ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like), Common Object Request Broker Architecture (CORBA), process pipes, shared files, and/or the like. Messages sent between discrete module components for inter-application communication or within memory spaces of a singular module for intra-application communication may be facilitated through the creation and parsing of a grammar. A grammar may be developed by using standard development tools such as lex, yacc, and/or the like, which allow for grammar generation and parsing functionality, which in turn may form the basis of communication messages within and between modules. Again, the configuration will depend upon the context of system deployment.

Tunneling Client Access Point Controller

FIG. **10** illustrates one embodiment incorporated into a tunneling client access point (TCAP) controller **1001**. Much of the description of the TCAPS of FIG. **9** applies to the TCAP, and as such, the disclosure focuses more upon the variances exhibited in the TCAP. In this embodiment, the TCAP controller **1001** may serve to process, store, search, identify, instruct, generate, match, and/or update data within itself, at a TCAPS, and/or through an AT.

The first and foremost difference between the TCAP and the TCAPS is that the TCAP is very small as was shown **130** of FIG. **1**. The TCAP may be packaged in plugin sticks, often, smaller than the size of a human thumb. In one embodiment, a TCAP may be hardened for military use. In such an embodiment, the shell **1001** may be composed of metal, and/or other durable composites. Also, components within may be shielded from radiation.

In one embodiment, the TCAP controller **1001** may be connected to and/or communicate with entities such as, but not limited to: one or more users from an access terminal **1011***b*. The access terminal itself may be connected to peripherals such as user input devices (e.g., keyboard **1012***a*, mouse **1012***b*, etc.); and/or a communications network **1013** in manner similar to that described in FIG. **9**.

A TCAP controller **1001** may be based on common computer systems components that may comprise, but are not limited to, components such as: a computer systemization **1002** connected to memory **1029**. Optionally, the TCAP controller **1001** may convey information **1058**, produce output through an output device **1048**, and obtain input from control device **1018**.

Control Device

The control device **1018** may be optionally provided to accept user input to control access to the TCAP controller. In one embodiment, the control device may provide a keypad **1028**. Such a keypad would allow the user to enter passwords, personal identification numbers (PIN), and/or the like.

In an alternative embodiment, the control device may include a security device **1038**. In one embodiment, the security device is a fingerprint integrated circuit (fingerprint IC) that provides biometric fingerprint information such as, but not limited to AuthenTec Inc.'s FingerLoc™ AF-S2™. Either a fingerprint IC and/or other biometric device will provide biometric validation information that may be used to confirm the identity of a TCAP user and ensure that transactions are legitimate. In alternative embodiments, a simple button, heat sensor, and/or other type of user input functionality may be provided solely and/or in concert with other types of control device types. The control device may be connected to the I/O interface, the system bus, or the CPU directly.

The output device **1048** is used to provide status information to the user. In one alternative embodiment, the output device is an LCD panel capable of providing alpha numeric and/or graphic displays. In an alternative embodiment, the output device may be a speaker providing audible signals indicating errors and/or actually streaming information that is audible to the user, such as voice alerts. The output device may be connected to the I/O interface, the system bus, or the CPU directly.

The conveyance information **1058** component of the TCAP controller may include any number of indicia representing the TCAP's source on the cover **1001**. Source conveying indicia may include, but is not limited to: an owner name **1059** for readily verifying a TCAP user; a photo of the owner **1060** for readily verifying a TCAP controller owner; mark designating the source that issued the TCAP **1061**, **1001** such as a corporate logo, and/or the like; fanciful design information **1062** for enhancing the visual appearance of the TCAP; and/or the like. It should be noted that the conveyance information **11421** may be positioned anywhere on the cover **1189**.

Computer Systemization

A computer systemization **1002** may comprise a clock **1030**, central processing unit (CPU) **1003**, a read only memory (ROM) **1006**, a random access memory (RAM) **1005**, and/or an interface bus **1007**, and most frequently, although not necessarily, are all interconnected and/or communicating through a system bus **1004**. Optionally the computer systemization may be connected to an internal power source **1086**. Optionally, a cryptographic processor **1026** may be connected to the system bus. The system clock typically has a crystal oscillator and provides a base signal. Of course, any of the above components may be connected directly to one another, connected to the CPU, and/or organized in numerous variations employed as exemplified by various computer systems.

The CPU comprises at least one low-power data processor adequate to execute program modules for executing user and/or system-generated requests. The CPU may be a microprocessor such as ARM's Application Cores, Embedded Cores, Secure Cores; Motorola's DragonBall; and/or the like processor(s).

Power Source

The power source **1086** may be of any standard form for powering small electronic circuit board devices such as but not limited to: alkaline, lithium hydride, lithium ion, nickel cadmium, solar cells, and/or the like. In the case of solar cells, the case provides an aperture through which the solar cell protrudes are to receive photonic energy. The power cell **1086** is connected to at least one of the interconnected subsequent components of the TCAP thereby providing an electric current to all subsequent components. In one example, the power cell **1086** is connected to the system bus component **1004**. In an alternative embodiment, an outside power source **1086** is provided through a connection across the I/O **1008** interface. For example, a USB and/or IEEE 1394 connection carries both data and power across the connection and is therefore a suitable source of power.

Interface Adapters

Interface bus(ses) **1007** may accept, connect, and/or communicate to a number of interface adapters, conventionally although not necessarily in the form of adapter cards, such as but not limited to: input output interfaces (I/O) **1008**, storage interfaces **1009**, network interfaces **1010**, and/or the like. Optionally, cryptographic processor interfaces **1027** similarly may be connected to the interface bus. The interface bus provides for the communications of interface adapters with one another as well as with other components of the computer systemization. Interface adapters are adapted for a compatible interface bus. In one embodiment, the interface bus provides I/O **1008** via a USB port. In an alternative embodiment, the interface bus provides I/O via an IEEE 1394 port. In an alternative embodiment, wireless transmitters are employed by interfacing wireless protocol integrated circuits (ICs) for I/O via the interface bus **1007**.

Storage interfaces **1009** may accept, communicate, and/or connect to a number of storage devices such as, but not limited to: storage devices **1014**, removable disc devices, and/or the like. Storage interfaces may employ connection protocols such as, but not limited to a flash memory connector, and/or the like. In one embodiment, an optional network interface may be provide **1010**.

Input Output interfaces (I/O) **1008** may accept, communicate, and/or connect to an access terminal **1011***b*. I/O may employ connection protocols such as, but not limited to: Apple Desktop Bus (ADB); Apple Desktop Connector (ADC); IEEE 1394a-b; infrared; PC AT; PS/2; parallel; radio; serial; USB, and/or the like; wireless component; and/or the like.

Wireless Component

In one embodiment a wireless component may comprise a Bluetooth chip disposed in communication with a transceiver **1043** and a memory **1029** through the interface bus **1007** and/or system bus **1004**. The transceiver may be either external to the Bluetooth chip, or integrated within the Bluetooth chip itself. The transceiver is a radio frequency (RF) transceiver operating in the range as required for Bluetooth transmissions. Further, the Bluetooth chip **1044** may integrate an input/output interface (I/O) **1066**. The Bluetooth chip and its I/O may be configured to interface with the TCAP controller through the interface bus, the system buss, and/or directly with the CPU. The I/O may be used to interface with other components such as an access terminal **1011***b* equipped with similar wireless capabilities. In one embodiment, the TCAP may optionally interconnect wirelessly with a peripheral device **912** and/or a control device **911** of FIG. **9**. In one example embodiment, the I/O may be based on serial line technologies, a universal serial bus (USB) protocol, and/or the like. In an alternative embodiment, the I/O may be based on the ISO 7816-3 standard. It should be noted that the Bluetooth chip in an alternative embodiment may be replaced with an IEEE 802.11b wireless chip. In another embodiment, both a Bluetooth chip and an IEEE 802.11b wireless chip may be used to communicate and or bridge communications with respectively enabled devices. It should further be noted that the transceiver **1043** may be used to wirelessly communicate with other devices powered by Bluetooth chips and/or IEEE 802.11b chips and/or the like. The ROM can provide a basic instruction set enabling the Bluetooth chip to use its I/O to communicate with other components. A number of Bluetooth chips are commercially available, and may be used as a Bluetooth chip in the wireless component, such as, but not limited to, CSR's BlueCore line of chips. If IEEE 802.11b functionality is required, a number of chips are commercially available for the wireless component as well.

Cryptographic units such as, but not limited to, microcontrollers, processors **1026**, and/or interfaces **1027** may be attached, and/or communicate with the TCAP controller. A Secure Core component commonly manufactured by ARM, Inc. and may be used for and/or within cryptographic units.

Memory

Generally, any mechanization and/or embodiment allowing a processor to affect the storage and/or retrieval of information is regarded as memory **1029**. However, memory is a fungible technology and resource, thus, any number of memory embodiments may be employed in lieu of or in concert with one another. It is to be understood that a TCAP controller and/or a computer systemization may employ various forms of memory **1029**. In a typical configuration, memory **1029** will include ROM **1006**, RAM **1005**, and a storage device **1014**. A storage device **1014** may be any conventional computer system storage. Storage devices may include flash memory, micro hard drives, and/or the like.

Module Collection

The memory **1029** may contain a collection of program and/or database modules and/or data such as, but not limited to: operating system module(s) **1015** (operating system); information server module(s) **1016** (information server); user interface module(s) **1017** (user interface); Web browser module(s) **1018** (Web browser); database(s) **1019**; cryptographic server module(s) **1020** (cryptographic server); access terminal module **1021**; TCAP module(s) **1035**; and/or the like (i.e., collectively a module collection). These modules may be stored and accessed from the storage devices and/or from storage devices accessible through an interface bus. Although non-conventional software modules such as those in the module collection, typically, are stored in a local storage device **1014**, they may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote storage facilities through an access terminal, communications network, ROM, various forms of memory, and/or the like. In one embodiment, all data stored in memory is encrypted by employing the cryptographic server **1020** as described in further detail below. In one embodiment, the ROM contains a unique TCAP identifier. For example, the TCAP may contain a unique digital certificate, number, and/or the like, which may be used for purposes of verification and encryption across a network and/or in conjunction with a TCAPS.

Operating System

The operating system module **1015** is executable program code facilitating the operation of a TCAP controller. Typically, the operating system facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or the like. The operating system may be a highly fault tolerant, scalable, and secure system such as Linux, and/or the like operating systems. However, more limited and/or less secure operating systems also may be employed such as Java runtime OS, and/or the like. An operating system may communicate to and/or with other modules in a module collection, including itself, and/or the like. Most frequently, the operating system communicates with other program modules, user interfaces, and/or the like. For example, the operating system may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. The operating system, once executed by the CPU, may enable the interaction with an access terminal, communications networks, data, I/O, peripheral devices, program modules, memory, user input devices, and/or the like. The operating system may provide communications protocols that allow the TCAP controller to communicate with other entities through an access terminal. Various communication protocols may be used by the TCAP controller as a subcarrier transport mechanism for interaction, such as, but not limited to: TCP/IP, USB, and/or the like.

Information Server

An information server module **1016** is stored program code that is executed by the CPU. The information server may be a conventional Internet information server such as, but not limited to Apache Software Foundation's Apache, and/or the like. The information server may allow for the execution of program modules through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective−) C (++), Common Gateway Interface (CGI) scripts, Java, JavaScript, Practical Extraction Report Language (PERL), Python, WebObjects, and/or the like. The information server may support secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), and/or the like. The information server provides results in the form of Web pages to Web browsers, and allows for the manipulated generation of the Web pages through interaction with other program modules. An information server may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the information server communicates with the TCAP database **1019**, operating systems, other program modules, user interfaces, Web browsers, and/or the like.

Access to TCAP database may be achieved through a number of database bridge mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through inter-application communication channels as enumerated below (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed through the bridge mechanism into appropriate grammars as required by the TCAP. In one embodiment, the information server would provide a Web form accessible by a Web browser. Entries made into supplied fields in the Web form are tagged as having been entered into the particular fields, and parsed as such. The entered terms are then passed along with the field tags, which act to instruct the parser to generate queries directed to appropriate tables and/or fields. In one embodiment, the parser may generate queries in standard SQL by instantiating a search string with the proper join/select commands based on the tagged text entries, wherein the resulting command is provided over the bridge mechanism to the TCAP as a query. Upon generating query results from the query, the results are passed over the bridge mechanism, and may be parsed for formatting and generation of a new results Web page by the bridge mechanism. Such a new results Web page is then provided to the information server, which may supply it to the requesting Web browser.

Also, an information server may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

User Interface

A user interface module **1017** is stored program code that is executed by the CPU. The user interface may be a conventional graphic user interface as provided by, with, and/or atop operating systems and/or operating environments such as Apple Macintosh OS, e.g., Aqua, Microsoft Windows (NT/XP), Unix X Windows (KDE, Gnome, and/or the like), and/or the like. The TCAP may employ code natively compiled for various operating systems, or code compiled using Java. The user interface may allow for the display, execution, interaction, manipulation, and/or operation of program modules and/or system facilities through textual and/or graphical facilities. The user interface provides a facility through which users may affect, interact, and/or operate a computer system. A user interface may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the user interface communicates with operating systems, other program modules, and/or the like. The user interface may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Web Browser

A Web browser module **1018** is stored program code that is executed by the CPU. A small-scale embedded Web browser may allow the TCAP to access and communicate with an attached access terminal, and beyond across a communications network. An example browser is Blazer, Opera, FireFox, etc. A browsing module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. Of course, in place of a Web browser and information server, a combined application may be developed to perform similar functions of both. The combined application would similarly affect the obtaining and the provision of information to users, user agents, and/or the like from TCAP enabled nodes. The combined application may be nugatory on systems employing standard Web browsers.

TCAP Database

A TCAP database module **1019** may be embodied in a database and its stored data. The database is stored program code, which is executed by the CPU; the stored program code portion configuring the CPU to process the stored data. In one embodiment, the TCAP database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. If the TCAP database is implemented as a data-structure, the use of the TCAP database may be integrated into another module such as the TCAP module. Databases may be consolidated and/or distributed in countless variations through standard data processing techniques. Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized and/or integrated. In one embodiment, the database module **1019** includes three tables **1019***a-c*. A user accounts table **1019***a* includes fields such as, but not limited to: a user name, user address, user authorization information (e.g., user name, password, biometric data, etc.), user credit card, organization, organization account, TCAP unique identifier, account creation data, account expiration date; and/or the like. In one embodiment, user accounts may be activated only for set amounts of time and will then expire once a specified date has been reached. An user data table **1019***b* includes fields such as, but not limited to: a TCAP unique identifier, backup image, data store, organization account, and/or the like. In one embodiment, the entire TCAP memory **1029** is processes into an image and spooled to a TCAPS for backup storage. A user programs table **1019***c* includes fields such as, but not limited to: system programs, organization programs, programs to be synchronized, and/or the like. It should be noted that any unique fields may be designated as a key field throughout. In an alternative embodiment, these tables have been decentralized into their own databases and their respective database controllers (i.e., individual database controllers for each of the above tables). Employing standard data processing techniques, one may further distribute the databases over several computer systemizations and/or storage devices. Similarly, configurations of the decentralized database controllers may be varied by consolidating and/or distributing the various database modules **1019***a-c*. The TCAP may be configured to keep track of various settings, inputs, and parameters via database controllers.

A TCAP database may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAP database communicates with a TCAP module, other program modules, and/or the like. The database may contain, retain, and provide information regarding other nodes and data.

Cryptographic Server

A cryptographic server module **1020** is stored program code that is executed by the CPU **1003**, cryptographic processor **1026**, cryptographic processor interface **1027**, and/or the like. Cryptographic processor interfaces will allow for expedition of encryption and/or decryption requests by the cryptographic module; however, the cryptographic module, alternatively, may run on a conventional CPU. The cryptographic module allows for the encryption and/or decryption of provided data. The cryptographic module allows for both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or decryption. The cryptographic module may employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital signatures, dual signatures, enveloping, password access protection, public key management, and/or the like. The cryptographic module will facilitate numerous (encryption and/or decryption) security protocols such as, but not limited to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like. The cryptographic module facilitates the process of "security authorization" whereby access to a resource is inhibited by a security protocol wherein the cryptographic module effects authorized access to the secured resource. In addition, the cryptographic module may provide unique identifiers of content, e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A cryptographic module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. The cryptographic module supports encryption schemes allowing for the secure transmission of information across a communications network to enable a TCAP module to engage in secure transactions if so desired. The cryptographic module facilitates the secure accessing of resources on TCAP and facilitates the access of secured resources on remote systems; i.e., it may act as a client and/or server of secured resources. Most frequently, the cryptographic module communicates with information servers, operating systems, other program modules, and/or the like. The cryptographic module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses. In one embodiment, the TCAP employs the cryptographic server to encrypt all data stored in memory **1029** based on the TCAP's unique ID and user's authorization information. In another embodiment, the TCAP employs the cryptographic server to encrypt all data sent through the access terminal based in the TCAP's unique ID and user's authorization information.

TCAP

A TCAP module **1035** is stored program code that is executed by the CPU. The TCAP affects accessing, obtaining and the provision of information, services, storage, transactions, and/or the like within its memory and/or across various communications networks. The TCAP enables users to simply access data and/or services from any location where an access terminal is available. It provides secure, extremely low powerful and ultra portable access to data and services that were heretofore impossible. The TCAP coordinates with the TCAP database to identify interassociated items in the generation of entries regarding any related information. A TCAP module enabling access of information between nodes may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective–) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the TCAP server employs a cryptographic server to encrypt and decrypt communications. A TCAP module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the TCAP module communicates with a TCAP database, a TCAP access terminal module **1021** running on an access terminal **1011***b*, operating systems, other program modules, and/or the like. The TCAP may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Access Terminal Module

An access terminal module **1021** is stored program code that is executed by a CPU. In one embodiment, the TCAP allows the access terminal **1011***b* to access its memory **1029** across its I/O **1008** and the access terminal executes the module. The access terminal module affects accessing, obtaining and the provision of information, services, storage, transactions, and/or the like within the TCAP's and access terminal's memory and/or across various communications networks. The access terminal module **1021** acts as a bridge through which the TCAP can communicate with communications network, and through which users may interact with the TCAP by using the I/O of the access terminal. The access terminal module coordinates with the TCAP module **1035** to send data and communications back and forth. A access terminal module enabling access of information between the TCAP and access terminal may be developed by employing standard development tools such as, but not limited to: (ANSI) (Objective–) C (++), Apache modules, binary executables, Java, Javascript, mapping tools, procedural and object oriented development tools, PERL, Python, shell scripts, SQL commands, web application server extensions, WebObjects, and/or the like. In one embodiment, the access terminal module is compiled for target access terminal platform, e.g., for Windows. In an alternative embodiment, a processor independent approach is taken, e.g., Java is used, so that the access terminal module will run on multiple platforms. In another embodiment, the TCAP server employs a cryptographic server to encrypt and decrypt communications as between it, the TCAP, and outside servers. A access terminal module may communicate to and/or with other modules in a module collection, including itself, and/or facilities of the like. Most frequently, the access terminal module communicates with a TCAP, other program modules, and/or the like. The access terminal module may contain, communicate, generate, obtain, and/or provide program module, system, user, and/or data communications, requests, and/or responses.

Distributed TCAP

The structure and/or operation of any of the TCAP node controller components may be combined, consolidated, and/or distributed in any number of ways to facilitate development and/or deployment. Similarly, the module collection may be combined in any number of ways to facilitate deployment and/or development. To accomplish this, one may integrate the components into a common code base or in a facility that can dynamically load the components on demand in an integrated fashion.

The module collection may be consolidated and/or distributed in countless variations through standard data processing and/or development techniques. Multiple instances of any one of the program modules in the program module collection may be instantiated on a single node, and/or across numerous nodes to improve performance through load-balancing and/or data-processing techniques. Furthermore, single instances may also be distributed across multiple controllers and/or storage devices; e.g., databases. All program module instances and controllers working in concert may do so through standard data processing communication techniques.

The configuration of the TCAP controller will depend on the context of system deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of the underlying hardware resources may affect deployment requirements and configuration. Regardless of if the configuration results in more consolidated and/or integrated program modules, results in a more distributed series of program modules, and/or results in some combination between a consolidated and distributed configuration, data may be communicated, obtained, and/or provided. Instances of modules consolidated into a common code base from the program module collection may communicate, obtain, and/or provide data. This may be accomplished through intra-application data processing communication techniques such as, but not limited to: data referencing (e.g., pointers), internal messaging, object instance variable communication, shared memory space, variable passing, and/or the like.

If module collection components are discrete, separate, and/or external to one another, then communicating, obtaining, and/or providing data with and/or to other module components may be accomplished through inter-application data processing communication techniques such as, but not limited to: Application Program Interfaces (API) information passage; (distributed) Component Object Model ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like), Common Object Request Broker Architecture (CORBA), process pipes, shared files, and/or the like. Messages sent between discrete module components for inter-application communication or within memory spaces of a singular module for intra-application communication may be facilitated through the creation and parsing of a grammar. A grammar may be developed by using standard development tools such as lex, yacc, and/or the like, which allow for grammar generation and parsing functionality, which in turn may form the basis of communication messages within and between modules. Again, the configuration will depend upon the context of system deployment.

The entirety of this disclosure (including the Cover Page, Title, Headings, Field, Background, Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract, Figures, and otherwise) shows by way of illustration various embodiments in which the claimed inventions may be practiced. The advantages and features of the disclosure are of a representative sample of embodiments only, and are not exhaustive and/or exclusive. They are presented only to assist in understanding and teach the claimed principles. It should be understood that they are not representative of all claimed inventions. As such, certain aspects of the disclosure have not been discussed herein. That alternate embodiments may not have been presented for a specific portion of the invention or that further undescribed alternate embodiments may be available for a portion is not to be considered a disclaimer of those alternate embodiments. It will be appreciated that many of those undescribed embodiments incorporate the same principles of the invention and others are equivalent. Thus, it is to be understood that other embodiments may be utilized and functional, logical, organizational, structural and/or topological modifications may be made without departing from the scope and/or spirit of the disclosure. As such, all examples and/or embodiments are deemed to be non-limiting throughout this disclosure. Also, no inference should be drawn regarding those embodiments discussed herein relative to those not discussed herein other than for purposes of space and reducing repetition. For instance, it is to be understood that the logical and/or topological structure of any combination of any program modules (a module collection), other components and/or any present feature sets as described in the figures and/or throughout are not limited to a fixed operating order and/or arrangement, but rather, any disclosed order is exemplary and all equivalents, regardless of order, are contemplated by the disclosure. Furthermore, it is to be understood that such features are not limited to serial execution, but rather, any number of threads, processes, services, servers, and/or the like that may execute asynchronously, simultaneously, synchronously, and/or the like are contemplated by the disclosure. As such, some of these features may be mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some features are applicable to one aspect of the invention, and inapplicable to others. In addition, the disclosure includes other inventions not presently claimed. Applicant reserves all rights in those presently unclaimed inventions including the right to claim such inventions, file additional applications, continuations, continuations in part, divisions, and/or the like thereof. As such, it should be understood that advantages, embodiments, examples, functional, features, logical, organizational, structural, topological, and/or other aspects of the disclosure are not to be considered limitations on the disclosure as defined by the claims or limitations on equivalents to the claims.

What is claimed is:

1. A portable device configured to communicate with a terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the portable device comprising:

(a) an external communication interface configured to enable the transmission of communications between the portable device and the terminal;

(b) a processor; and

(c) a memory having executable program code stored thereon, including:

(1) third program code which, when executed by the portable device processor, is configured to provide a communications node on the portable device to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and facilitate communications to the terminal and to a communications network node through the terminal network communication interface; and

(2) fourth program code which is configured to be executed by the portable device processor in response to a communication received by the portable device resulting from user interaction with the interactive user interface;

wherein the portable device is configured to facilitate communications through the communication node on the terminal and the terminal network interface to a communications network node.

2. The portable device according to claim **1**, wherein the fourth program code which, when executed by the portable device processor, is configured to cause a communication to be transmitted to the communication network node.

3. The portable device according to claim **2**, wherein the communication caused to be transmitted to the communication network node facilitates verification of the portable device.

4. The portable device according to claim **2**, wherein the communication caused to be transmitted to the communication network node facilitates the transmission of encrypted communications from the communication network node to the terminal.

5. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates access to the communication network node database.

6. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates the download of content on the communication network node database to the terminal.

7. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates the download of program code on the communication network node to the terminal.

8. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates the upload of content on to the communication network node database.

9. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates the upload of program code on to the communication network node database.

10. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates synchronizing content on the portable device with content on the communication network node database.

11. The portable device according to claim **2**, wherein the communication network node comprises a database and the communication caused to be transmitted to the communication network node facilitates the download of a live data feed to the terminal.

12. The portable device according to claim **11**, wherein the live data feed is presented on the terminal output component.

13. The portable device according to claim **1**, wherein the portable device is further configured to affect the presentation of the interactive user interface on the terminal output device.

14. The portable device according to claim **13**, wherein the portable device memory has fifth program code stored thereon, which, when executed, is configured to affect the presentation of the interactive user interface on the terminal output device.

15. The portable device according to claim **13**, wherein the portable device is configured to provide the terminal with access to content stored on the portable device memory which affects the presentation of the interactive user interface on the terminal output device.

16. The portable device according to claim **15**, wherein the content comprises a user name and/or a portable device identifier, and the interactive user interface is affected to present the user name and/or portable device identifier.

17. The portable device according to claim **1**, wherein the second program code is stored on the portable device memory and the portable device is configured to provide the terminal with access to the second program code.

18. The portable device according to claim **17**, wherein the portable device is configured to load the second program code on to the terminal.

19. The portable device according to claim **1**, wherein the portable device is configured to cause the terminal to execute the second program code and present an interactive user interface on the terminal output component.

20. The portable device according to claim **19**, wherein the portable device memory has fifth program code stored thereon, which, when executed by the portable device processor, is configured to cause the terminal to execute the second program code and present an interactive user interface on the terminal output component.

21. The portable device according to claim **1**, wherein the interactive user interface comprises a graphic user interface.

22. The portable device according to claim **1**, wherein the first program code is stored on the portable device memory and the portable device is configured to provide the terminal with access to the first program code.

23. The portable device according to claim **22**, wherein the portable device is configured to load the first program code on to the terminal.

24. The portable device according to claim **1**, wherein the portable device memory comprises one or more of the group consisting of flash memory, read only memory, random access memory, micro hard drives and the like.

25. The portable device according to claim **1**, wherein the communications network node comprises a server.

26. The portable device according to claim **1**, wherein the external communication interface comprises a universal serial bus interface.

27. The portable device according to claim **1**, wherein the external communication interface comprises a wireless communication interface.

28. A method implemented on a portable device comprising a processor, a memory having executable program code stored thereon, and an external communication interface for enabling the transmission of a plurality of communications between the portable device and a terminal comprising a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to the portable device and to a communications network node through the terminal network communication interface, the method comprising:

(a) causing the terminal to execute the first program code to present an interactive user interface on the terminal output component;

(b) executing third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal communication interface;

(c) executing fourth program code stored on the portable device memory in response to a communication received by the portable device resulting from user interaction with the interactive user interface; and

(d) facilitating a communication to be transmitted through the terminal network interface to a communications network node.

29. A system implementing a terminal having a processor, an input component, an output component, a network communication interface, and a memory configured to store executable program code, including first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component, and second program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal to facilitate communications to and from the terminal, the system comprising:

(a) a communications network node; and

(b) a portable device comprising an external communication interface for enabling the transmission of a plurality of communications between the portable device and the terminal, a processor, and a memory, wherein the memory has executable program code stored thereon, the portable device configured to:

(1) cause the terminal to execute the first program code to present an interactive user interface on the terminal output component;

(2) execute third program code stored on the portable device memory to provide a communications node on the portable device configured to coordinate with the communications node on the terminal and establish a communications link between the portable device and the terminal, and to facilitate communications to the terminal and to a communications network node through the terminal communication interface;

(3) execute fourth program code stored on the portable device memory in response to a communication received by the portable device resulting from user interaction with the interactive user interface; and

(4) facilitate a communication to be transmitted through the terminal network interface to a communications network node.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 9,059,969 B2	Page 1 of 1
APPLICATION NO.	: 13/960514	
DATED	: June 16, 2015	
INVENTOR(S)	: Scott McNulty	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims

In Claim 19, Column 32, Line 42, delete “second” and insert --first.--

In Claim 20, Column 32, Line 48, delete “second” and insert --first.--

Signed and Sealed this
Third Day of April, 2018

Andrei Iancu

Andrei Iancu
Director of the United States Patent and Trademark Office

EXHIBIT 12

FILED UNDER SEAL

Page 1

CONFIDENTIAL - ATTORNEYS' EYES - AVI RUBIN

UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IOENGINE, LLC,

Plaintiff/Counterclaim

Defendant

vs. Case No. 18-452-WCB

PAYPAL HOLDINGS, INC.,

Defendant/Counterclaim

Plaintiff.

January 18, 2022

9:28 a.m. - 5:54 p.m.

Bonita Springs, Florida

** CONFIDENTIAL - ATTORNEY'S EYES ONLY - SOURCE CODE **

REMOTE VIDEOTAPE DEPOSITION OF

AVIEL DAVID RUBIN

Taken on behalf of the Defendant/Counterclaim Plaintiff before Michael J. D'Amato, RMR, Notary Public in and for the State of Florida at Large, pursuant to Notice of Taking Deposition in the above cause.

Job # 205077

Page 2

CONFIDENTIAL - ATTORNEYS' EYES - AVI RUBIN

REMOTE APPEARANCES

For the Plaintiff/Counterclaim Defendant:

DECHERT
1095 Avenue of the Americas
New York, NY 10036
BY: GREGORY CHUEBON, ESQ.
LUKE REILLY, ESQ.

For the Defendant/Counterclaim Plaintiff:

ORRICK, HERRINGTON & SUTCLIFFE
1000 Marsh Road
Menlo Park, CA 94025
BY: JARED BOBROW, ESQ.

SUNSTEIN
100 High Street
Boston, MA 02110
BY: SHARONA STERNBERG, ESQ.
KERRY TIMBERS, ESQ.

ALSO PRESENT: PHILIP RIZZUTI, Videographer, TSG

Page 3

INDEX

Page 5

THE VIDEOGRAPHER: Good morning, counsel. My name is Phil Rizzuti. I am a legal videographer in association with TSG Reporting Inc. Due to the severity of the COVID 19 and following the practice of social distancing, I will not be in the same room with the witness. Instead, I will record this videotape deposition remotely. The reporter, Michael D'Amato, also will not be in the same room and will swear the witness remotely.

Do all parties stipulate to the validity of this video recording and remote swearing and that it will be admissible in the courtroom as if it had been taken following Rule 30 of the Federal Rules of Civil Procedure and the state's rules where this case is pending?

MR. BOBROW: This is Jared Bobrow for PayPal. Yes, we stipulate.

MR. CHUEBON: This is Greg Chuebon for Ioengine, and yes, we do so stipulate for the Ioengine V PayPal matter.

THE VIDEOGRAPHER: Thank you. This is the start of media labeled No. 1 of the video recorded deposition of Dr. Avi Rubin in the matter of Ioengine LLC v. PayPal Holdings, Inc. in the United

Page 6

States District Court for the District of Delaware, case No. 18-452-WCB.

This deposition is being held on January 18, 2022 at approximately 9:30 a.m. My name is Phil Ritz. I am the legal video specialist from TSG Reporting. The court reporter is Mike D'Amato in association with TSG Reporting. Counsel, please introduce yourself.

MR. BOBROW: Good morning. This is Jared Bobrow of the Orrick Herrington firm representing PayPal.

MR. CHUEBON: This is Greg Chuebon of Dechert representing Ioengine LLC. I'm joined today by my colleague, Luke Reilly also of Dechert.

THE VIDEOGRAPHER: Thank you. Will the court reporter please swear in the witness.

THEREUPON:

AVIEL DAVID RUBIN,

being by me first duly sworn or affirmed to tell the truth, the whole truth, and nothing but the truth, as hereinafter certified, responded and testified as follows:

THE WITNESS: I do.

THE REPORTER: Would you state your name and

Page 7

spell it for us?

THE WITNESS: My full name is Aviel, A-V-I-E-L, David, R-U-B-I-N and I go by Avi as a first name.

DIRECT EXAMINATION

BY MR. BOBROW:

Q. Good morning, Dr. Rubin.

A. Good morning.

Q. You are in Florida today, you said, is that right?

A. That's right.

Q. Are you in your home there or some other location?

A. I'm in a rental home that I rented for the week.

Q. I see. And is the background behind you on this video, is that the actual physical background?

A. No. That's a virtual background.

Q. What's that a virtual background of?

A. It looks like some kind of room with a sofa and some tall ceilings. I'm not exactly sure. I found it online.

Q. Okay. So it is not a physical location that you are familiar with, is that right?

Page 78

would be -- you can update a field in a record on a database.

Q. So for the '969 patent and the database that's claimed therein the database needs to be something that can be queried, can be searched, can be added to in terms of data, can be deleted to in terms of data and for which records and fields can be updated, is that right?

MR. CHUEBON: Objection to form.

A. It's saying that the data is structured in a way that enables those operations.

Q. And back in 2001 I take it the database is of the type that you described as very common?

MR. CHUEBON: Objection to form.

A. So relational databases were common and they had that format in 2001.

Q. Oracle, for example was a company that commercially sold databases, relational databases that had the functions that you described, correct?

MR. CHUEBON: Objection to form.

A. They did.

Q. And there were other companies, probably Sybase or others, back in the 2001 time frame that also provided relational databases of the type that you

Page 79

described, correct?

MR. CHUEBON: Objection to form.

A. I haven't looked at Sybase but in general those types of databases were common.

Q. Mr. McNulty didn't invent a new database, did he?

MR. CHUEBON: Objection to form.

A. He did not invent a new database technology. He did invent a way in which data can be synchronized to a database following the elements of the claim.

Q. Just in terms of the database itself, that's not something that he invented, correct?

MR. CHUEBON: Objection to form.

A. Databases existed before 2001.

Q. Now, with respect to the asserted claims you would agree with me that none of the asserted claims require that any of the claimed communications be encrypted, correct?

A. Can we look at a specific claim? I don't like to answer something about all of them together.

Q. Well let me ask, without looking at them, you can't answer the question of whether they do or don't require encryption?

MR. CHUEBON: Objection to form.

Page 216

CONFIDENTIAL - ATTORNEYS' EYES - AVI RUBIN

CERTIFICATE OF OATH

STATE OF FLORIDA)
SS
COUNTY OF PALM BEACH)

I, Michael J. D'Amato, Notary Public for the State of Florida, certify that on the 18th day of January 2022, AVIEL DAVID RUBIN personally appeared before me and was duly sworn or affirmed.

WITNESS my hand and official seal this 25th day of January 2022.

Michael J. D'Amato
Notary Public - State of Florida
My Commission # GG 960978
Expires: June 13, 2024

Page 217

CONFIDENTIAL - ATTORNEYS' EYES - AVI RUBIN

CERTIFICATE OF COURT REPORTER

I, MICHAEL J. D'AMATO, a Registered Merit Reporter and Notary Public in and for the State of Florida at Large, do HEREBY CERTIFY that I was authorized to and did stenographically report the deposition of AVIEL DAVID RUBIN; that a review of the transcript was requested; and that the foregoing transcript, pages from 1 to 216, is a true and accurate record of my stenographic notes.

I FURTHER CERTIFY that I am not a relative, employee, attorney, or counsel of any of the parties, nor am I a relative or employee of any of the parties' attorney or counsel connected with the action, nor am I financially interested in the action.

Dated this 25th day of January 2022.

MICHAEL J. D'AMATO,

Registered Merit Reporter

Dr. Aviel Rubin, PhD
IOENGINE, LLC v. PayPal Holdings Inc., 18-cv-452-WCB (D. Del)
January 18, 2022, Deposition Errata Sheet to Transcript

Page and Line No.	**Change**	**Reason for Change**
Passim	"Ioengine" to "IOENGINE"	Misspelling
18:19	"It's" to "She's"	Mis-transcription
37:8-9	"So I don't seem to have included that legal standard in my report." to "I summarize the standard in paragraph 168 of my report: whether or not Mr. McNulty had formed in his mind a definite and permanent idea of a complete and operative invention, as it was to be applied in practice"	Correction
38:15-17	"Since I'm not a lawyer and I didn't include that standard in this document, I can't map what I did to a specific legal document" to "I summarize the standard in paragraph 168 of my report: whether or not Mr. McNulty had formed in his mind a definite and permanent idea of a complete and operative invention, as it was to be applied in practice,"	Correction
42:2	"I don't recall the legal standard," to "I summarize the standard in paragraph 168 of my report: whether or not Mr. McNulty had formed in his mind a definite and permanent idea of a complete and operative invention, as it was to be applied in practice,"	Correction
50:23-24	"they describe more different communications" to "they describe a number of different communications"	Clarification
51:11-12	"It doesn't provide at the lowest level of synchronization a new protocol for synchronization." to "It doesn't provide, at the lowest level of synchronization, a new protocol for synchronization."	Mis-transcription
59:5	"would be" to "could be"	Clarification
60:18	"Yeah, I would say that's right." To "Yeah, if the other requirements of synchronization were met, for those two files I would say that's right"	Clarification

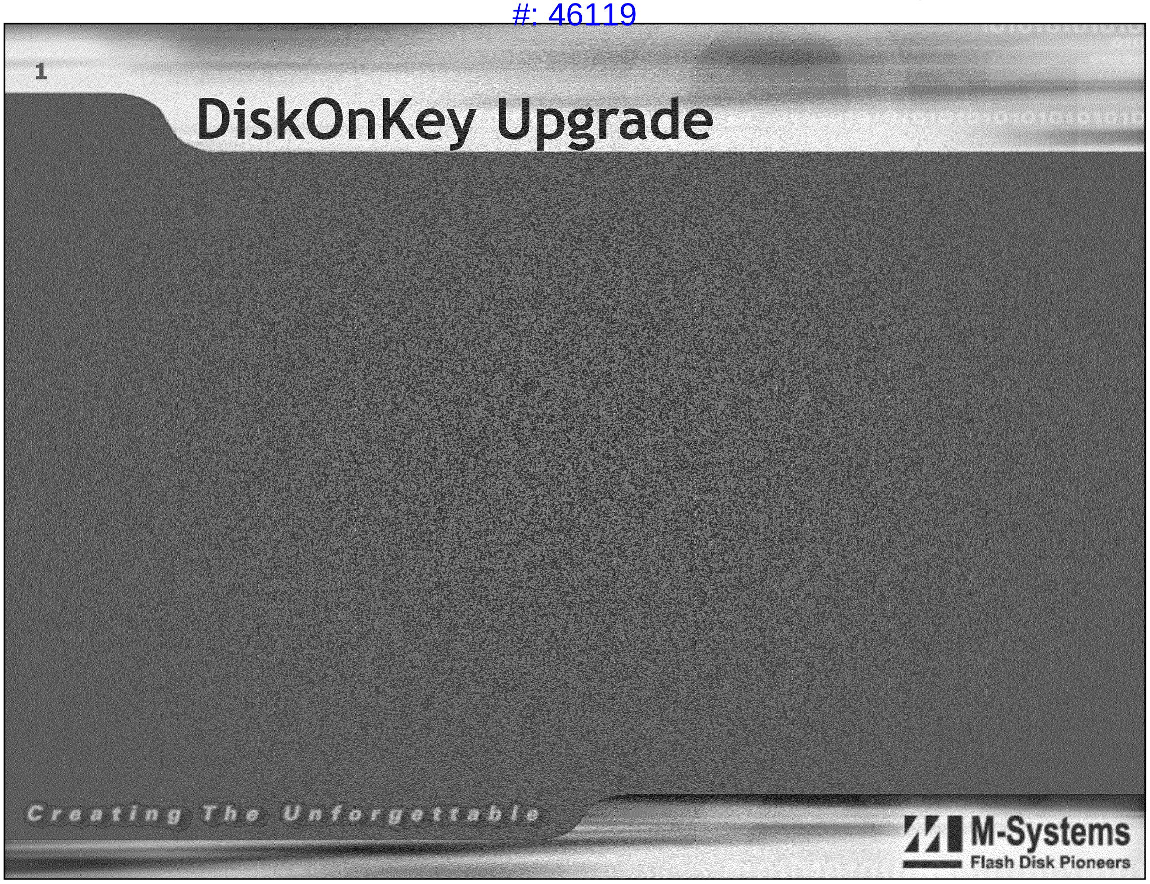
Page and Line No.	Change	Reason for Change
62:3	"So I was going to look at that citation." To "The citations here appear to be a typo, and there is no close brackets. But the '969 Patent describes this at 8:26-31, 9:32-27, and 12:29-31. Those should have been cited in paragraph 49 of my report."	Correction
67:17-18	"it would be an identifier information of that person" to "it would be an identifier information of that device"	Correction
68:25	"Right." To "Right, it says 'code on the portable device works it[s] way through the computer to the internet and contacts a web site and says Hi[,] I am here from a person[']s pocket."	Clarification
70:14	"the claim functionality of the portable device" to "the claimed functionality of the portable device"	Mis-transcription
76:5-6	"that the terminal in that spot is making a decision" to "that prevents the terminal in that spot from making a decision"	Clarification
77:4	"Yes." To "Yes, as a part of the environment."	Clarification
77:7	"Yes, that is my recollection, yes." To "Yes, at least as a part of the environment, that is my recollection, yes."	Clarification
81:7	"I agree with that." To "I agree with that, the communications are not limited to the payment context."	Clarification
83:13	"So cause is where you directly make it happen." To "So cause is where you directly or indirectly make it happen."	Clarification
83:18	"if I give you a baseball" to "if I give you a baseball without an established procedure of how it will be used"	Clarification
90:4	"board" to "cord"	Mis-transcription

Page and Line No.	Change	Reason for Change
91:12	“That’s not a requirement of the claims.” to “That’s not a requirement of the term ‘interactive user interface.’”	Clarification
108:11-12	“I was just applying my understanding of the word diligent.” To “I was just applying my understanding of the word diligent as counsel explained it to me.”	Clarification
114:13	“move this one to the text” to “use this one for the text”	Mis-transcription
116:10-11	“Those screens can contain prompts for -- yes, prompts.” To “Those screens can contain images of prompts for -- yes, images of prompts.”	Clarification
129:10	Remove “am I”	Mis-transcription
142:16	“That would make sense.” To “That is not what is disclosed.”	Correction
200:7-8	“I don't recall seeing that in the specification” to “I don't recall seeing anything that would require or exclude that in the specification”	Correction
202:16	“having some user ID and some secret” to “having some user ID and/or some secret”	Clarification

____2/14/2022________________

WITNESS' SIGNATURE DATE

EXHIBIT 13



1

DiskOnKey Upgrade

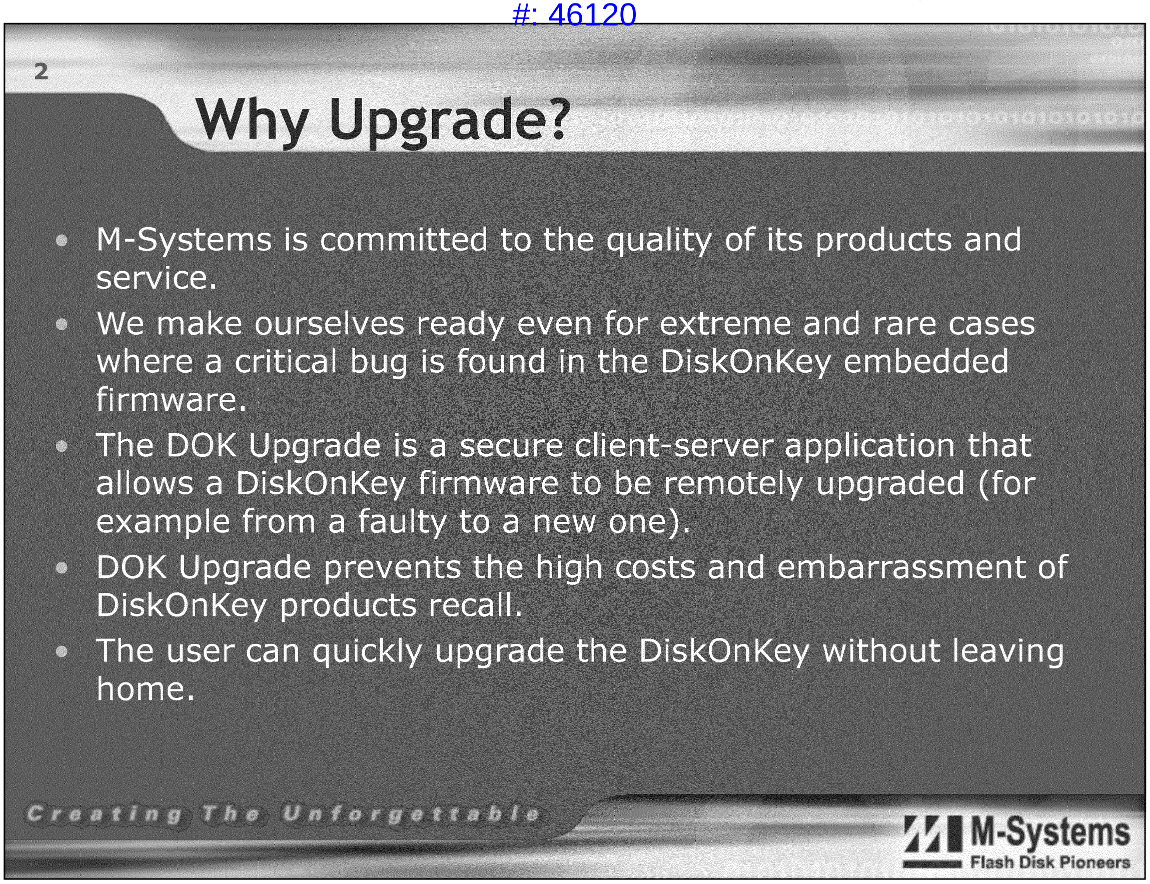
Creating The Unforgettable

M-Systems
Flash Disk Pioneers

Ingenico Inc. v. IOENGINE LLC
DX-336
C.A. No. 18-cv-00826-WCB

Confidential - Attorneys' Eyes Only

Exhibit
0024
Sobol



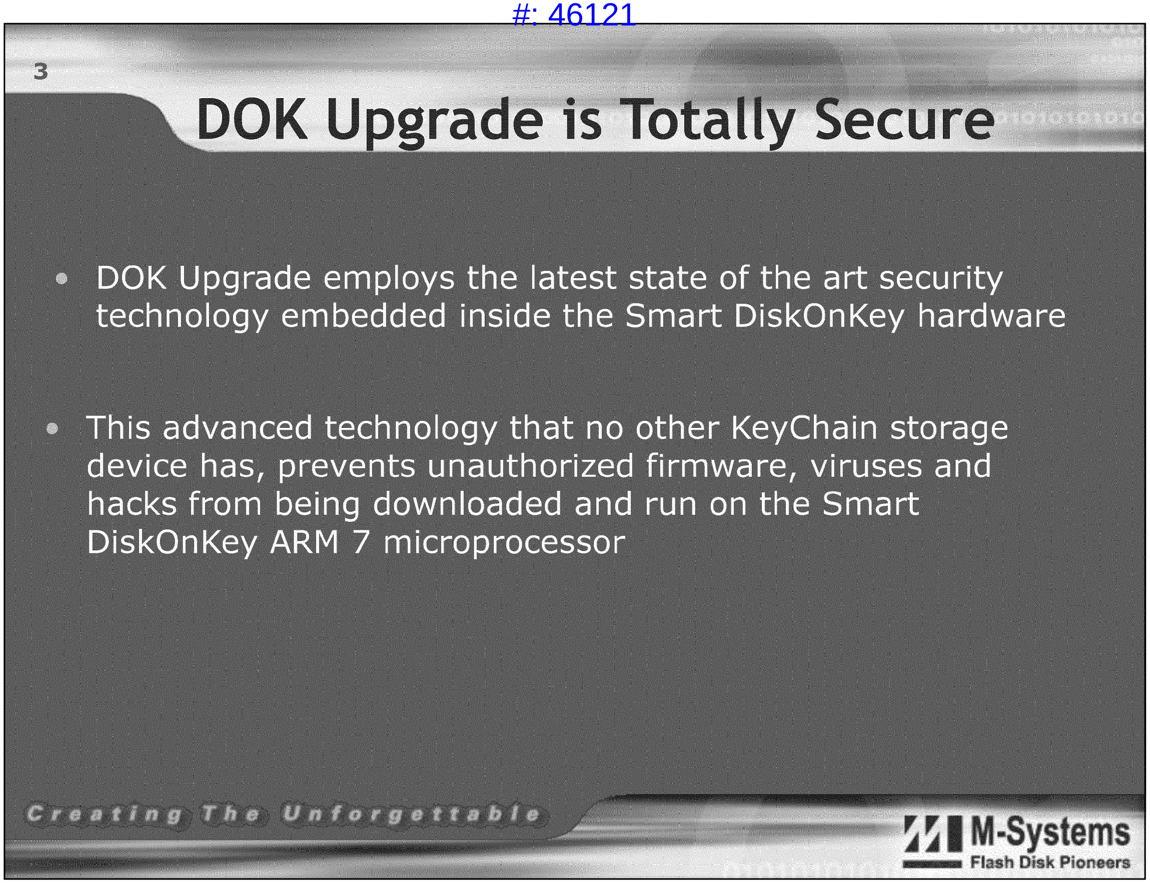
2

Why Upgrade?

- M-Systems is committed to the quality of its products and service.
- We make ourselves ready even for extreme and rare cases where a critical bug is found in the DiskOnKey embedded firmware.
- The DOK Upgrade is a secure client-server application that allows a DiskOnKey firmware to be remotely upgraded (for example from a faulty to a new one).
- DOK Upgrade prevents the high costs and embarrassment of DiskOnKey products recall.
- The user can quickly upgrade the DiskOnKey without leaving home.

Creating The Unforgettable

M-Systems
Flash Disk Pioneers



3

DOK Upgrade is Totally Secure

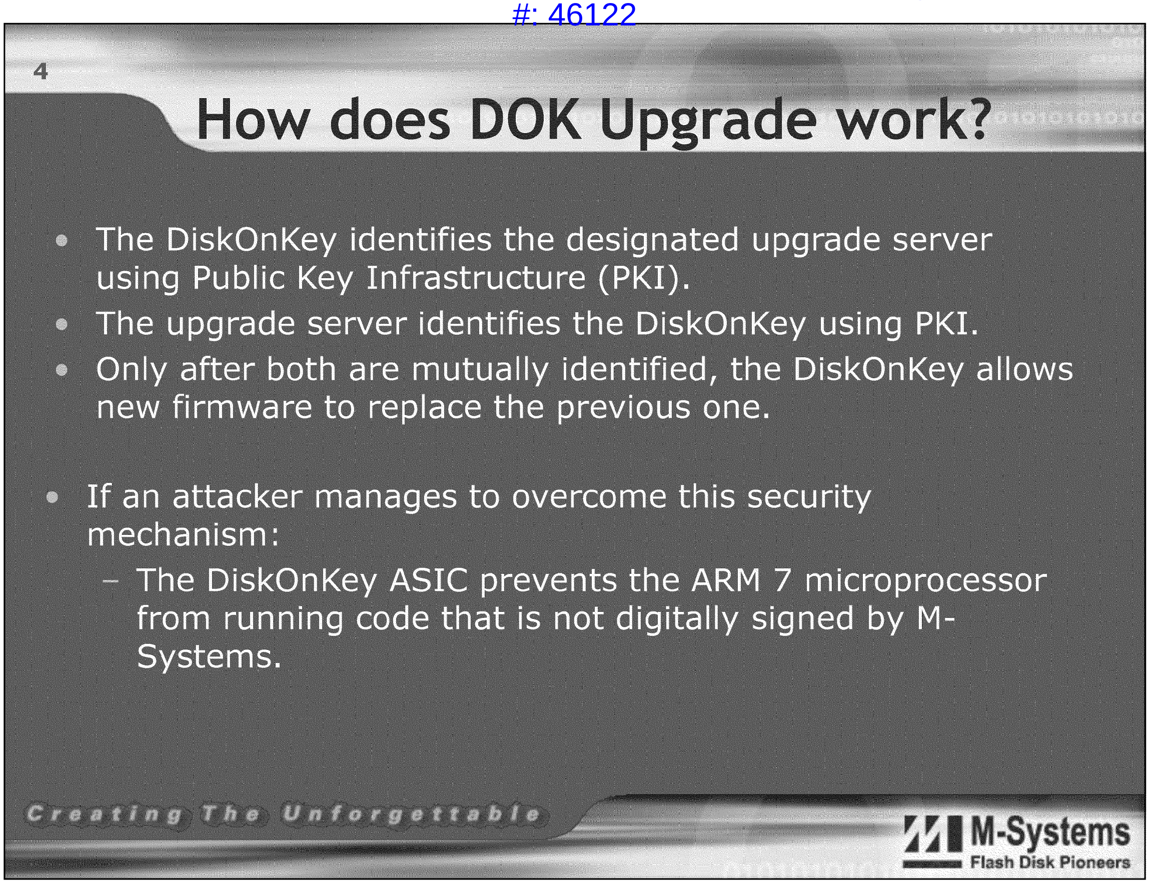
- DOK Upgrade employs the latest state of the art security technology embedded inside the Smart DiskOnKey hardware

- This advanced technology that no other KeyChain storage device has, prevents unauthorized firmware, viruses and hacks from being downloaded and run on the Smart DiskOnKey ARM 7 microprocessor

Creating The Unforgettable

M-Systems
Flash Disk Pioneers

DX-336.3



How does DOK Upgrade work?

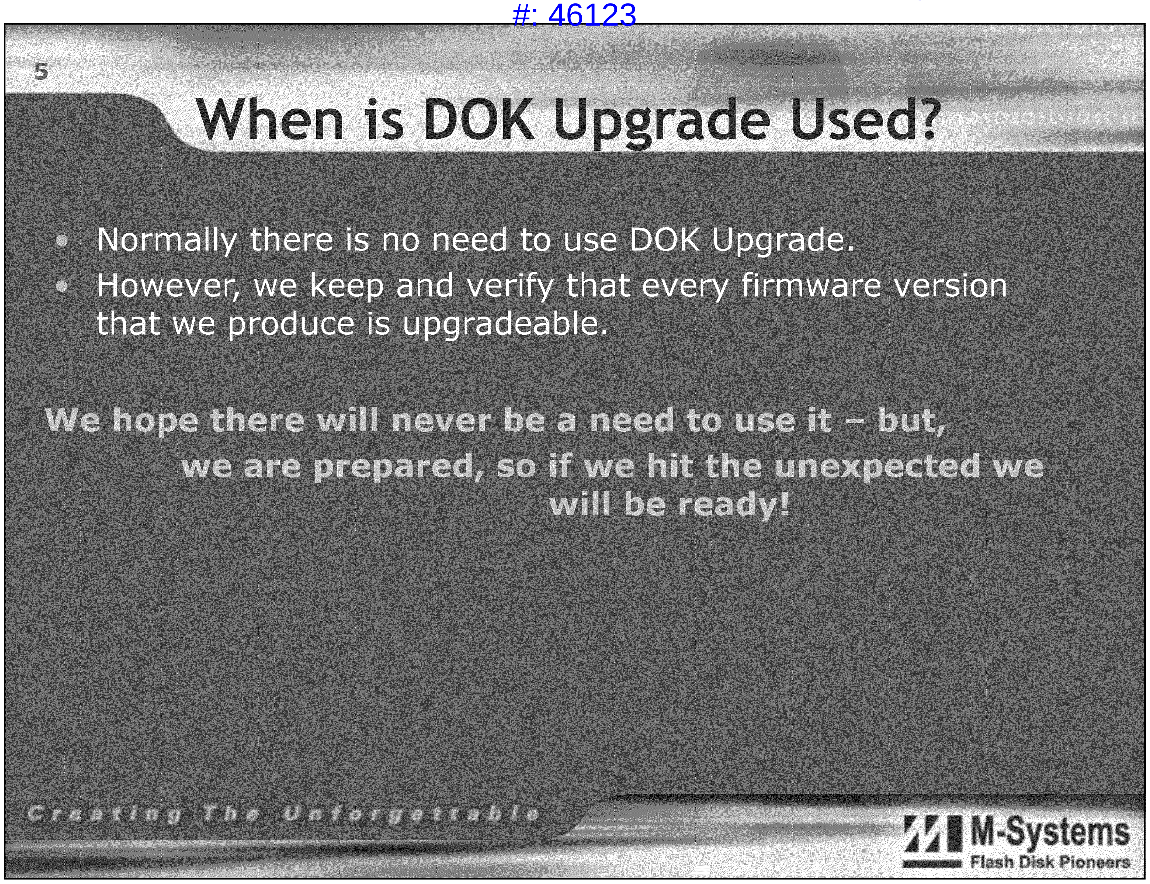
- The DiskOnKey identifies the designated upgrade server using Public Key Infrastructure (PKI).
- The upgrade server identifies the DiskOnKey using PKI.
- Only after both are mutually identified, the DiskOnKey allows new firmware to replace the previous one.

- If an attacker manages to overcome this security mechanism:
  - The DiskOnKey ASIC prevents the ARM 7 microprocessor from running code that is not digitally signed by M-Systems.

Creating The Unforgettable

M-Systems
Flash Disk Pioneers

DX-336.4



5

When is DOK Upgrade Used?

- Normally there is no need to use DOK Upgrade.
- However, we keep and verify that every firmware version that we produce is upgradeable.

We hope there will never be a need to use it – but, we are prepared, so if we hit the unexpected we will be ready!

Creating The Unforgettable

M-Systems
Flash Disk Pioneers

DX-336.5

Bates Beg :	WDC0012119
Bates End :	WDC0012123
Confidential Designation :	Confidential - Attorneys' Eyes Only
File Type :	Microsoft PowerPoint 97-2003 Presentation
Date and Time Last Modified :	3/22/2003 11:11:00 AM
Date and Time Created :	3/21/2003 4:48:00 AM
Email From :	
Email To :	
Email CC :	
Email BCC :	

EXHIBIT 14

M-Systems
Flash Disk Pioneers

DiskOnKey Software Development Kit

PRELIMINARY DOCUMENT

SEPTEMBER 2002

VER 1.5

Ingenico Inc. v. IOENGINE LLC
DX-333
C.A. No. 18-cv-00826-WCB

Exhibit
0018
Sobol

Confidential - Attorneys' Eyes Only

Contents

1 Overview

DiskOnKey is M-Systems' flash storage device that uses the USB port to download and transfer files between PCs, laptops and Macintosh computers without the need for drivers. M-Systems was the first in the industry to introduce this new keychain storage category, which quickly became an industry standard. M-Systems is now introducing another first: a Software Development Kit (SDK) for keychain storage. It enables software developers to write customized applications for the DiskOnKey, without facing the difficulties of handling the operating system (OS), the USB and the OS registry.

This document describes the system architecture on which M-Systems' DiskOnKey SDK is based, the library of functions that is available to software developers to activate the DiskOnKey and how to work with these functions. It also provides sample code and some troubleshooting tips.

1.1 Functionality

The DiskOnKey SDK binary library provides Application Proprietary Interfaces (APIs) that enable developers to customize applications by activating internal DiskOnKey mechanisms, based on M-Systems' core competence. Using DiskOnKey internal mechanisms not only saves application development time, but also ensures a smoother integration path, free of platform-related difficulties.

Software applications for the DiskOnKey are executed on a PC running Windows operating systems, including: Windows 98, Windows Me, Windows 2000 and Windows XP. The applications themselves can reside on the DiskOnKey.

DiskOnKey internal firmware contains passive functions and abilities, such as encryption and authentication, to secure data on the DiskOnKey ready to be activated by a software application.

1.2 Key Concepts

The DiskOnKey SDK is distributed as a Dynamic Linked Library (DLL), which is a Windows format for executables that are loaded to support software applications. This means that the developer does not need to integrate or compile unfamiliar source code.

The DiskOnKey SDK uses a set of raw APIs that are similar to the familiar Windows Win32 APIs. This set is written in C++ and in object-oriented programming (OOP) format. The APIs are simple and straightforward, using the same system that the OS uses to invoke the device. The APIs control basic device functionality, such as:

- Insertion, removal, safe ejection and formatting the DiskOnKey
- Multiple volumes
- Cookies
- Secure area
- Authentication and encryption

1.3 API Terminology

The following terms are used throughout the DiskOnKey SDK documentation.

1.3.1 Cookie

The **ICookie** interface enables an application to manage a cookie on the DiskOnKey. The cookie resides in the hidden (reserved) area on the DiskOnKey and is used to save integer and string information generated by an application, ready for future access. For example, the cookie can be used to save a password hint on the DiskOnKey. The cookie and hidden area are not directly accessible by the user.

1.3.2 Deposit

The **IDeposit** interface enables the user application to create a password-protected Safe area on the DiskOnKey, and to utilize the deposit capabilities.

The DiskOnKey provides a secured file-system area (disk) that can be accessed via the SDK API. This area, called the Deposit partition, is a part of the DiskOnKey flash memory, and is hidden from the operating system. It enables the DiskOnKey user to utilize a secured area inside the DiskOnKey.

To expose the Deposit area, you must insert the deposit password, via the SDK API. The DiskOnKey authenticates the password, and then exposes the Deposit area to the operating system.

1.3.3 Device Listening

Device listening is the mechanism for monitoring when a DiskOnKey is inserted or removed. Once the DiskOnKey device is connected, the application receives an event, and can then communicate with the device via the SDK's **CDOK** object.

1.4 SDK Contents

The SDK contains the following items:

- Static or dynamic libraries
- Header files
- Source and binary examples

1.5 Requirements and Features

The following sections detail the DiskOnKey requirements and features.

1.5.1 Operating Systems Supported

The DiskOnKey supports the following operating systems:

- Windows 98 SE
- Windows Me
- Windows 2000
- Windows XP

1.5.2 System Requirements

The system requirements for the DiskOnKey include:

- 64MB RAM
- Pentium II 266 MHz Processor
- MSVC 6.0. Refer to Building Applications for DiskOnKey.

1.5.3 DiskOnKey Features

The DiskOnKey features include:

- **DiskOnKey Capacities:** 8, 16, 32, 64, 128, 256 and 512MB
- **DiskOnKey Types:** Fixed, Removable

1.5.4 SDK Requirements

When using the current SDK version with Windows 2000 or Windows XP, the user must be logged in with administrator privileges in order to access disk devices, such as the DiskOnKey. The current privileges status can be checked using the `CDOK::canAccessDevice` method. This issue will be resolved in a future SDK version.

2 System Architecture

Figure 2-1 illustrates the interaction between the SDK objects, the application and the DiskOnKey.

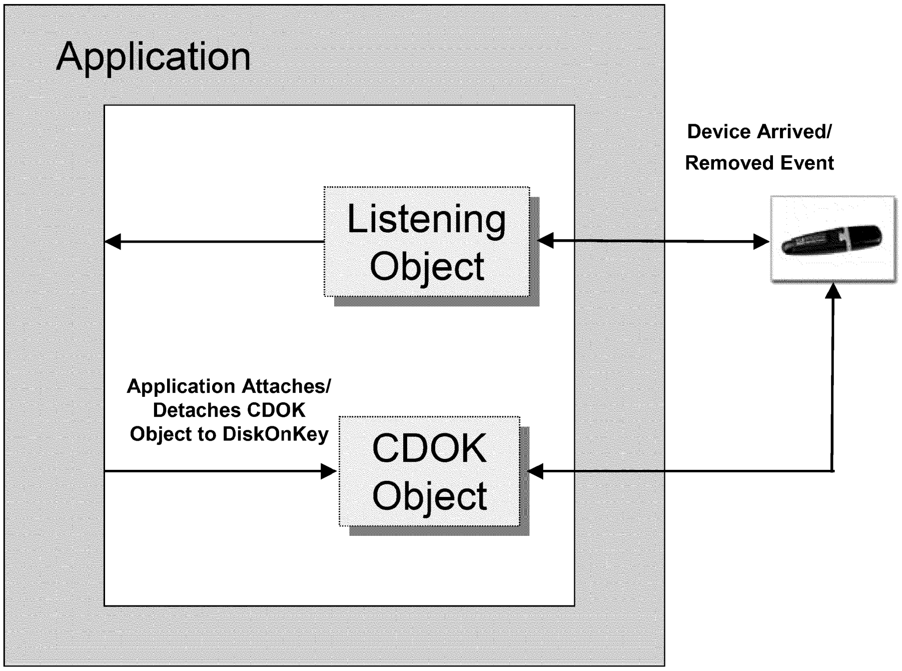


Figure 2-1: Device Listening

The components in this diagram are described in detail in the sections below:

- CDOK - DiskOnKey Device Interaction
- SDK Interfaces
- Device Listening
- DiskOnKey Structure

2.1 CDOK - DiskOnKey Device Interaction

The **CDOK** object enables an application to manage a DiskOnKey device by attaching it to a DiskOnKey device. More than one application **CDOK** object can be attached to a specific DiskOnKey device. Figure 2-2 illustrates the **CDOK** object interaction with the DiskOnKey devices.

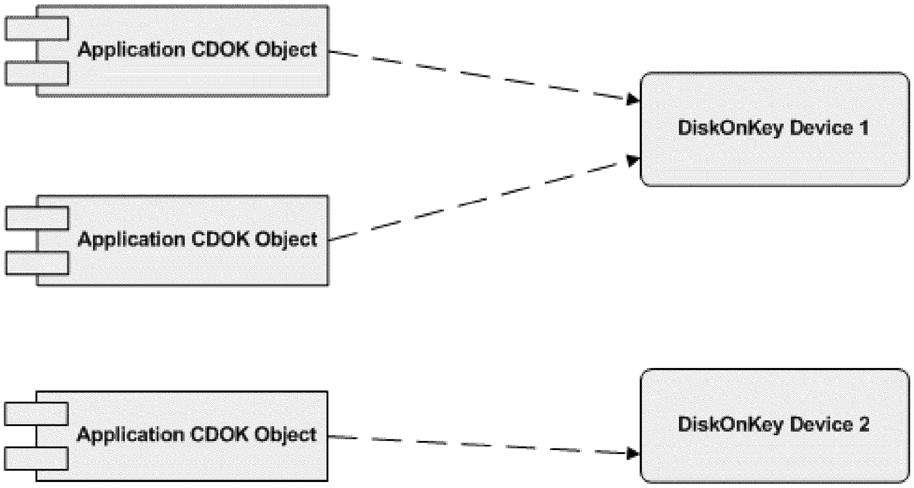


Figure 2-2: CDOK - DiskOnKey Interaction

The SDK provides methods to attach the **CDOK** object to a specific DiskOnKey device, and enables the **CDOK** object to work with that device, that is, run **CDOK** methods and APIs.

The SDK automatically detaches the **CDOK** object from the DiskOnKey if the device is ejected (or if the device is removed) and the SDK automatically changes the **CDOK** status to ‘not connected’. The **CDOK** object is not automatically re-attached if the DiskOnKey is plugged in. Every additional function call will return a **DOK_NOT_CONNECTED** status. The application must reattach the **CDOK** object to the DiskOnKey to in order to continue.

2.2 SDK Interfaces

Figure 2-3 illustrates the **CDOK** object interaction with the various interfaces provided by the DiskOnKey SDK.

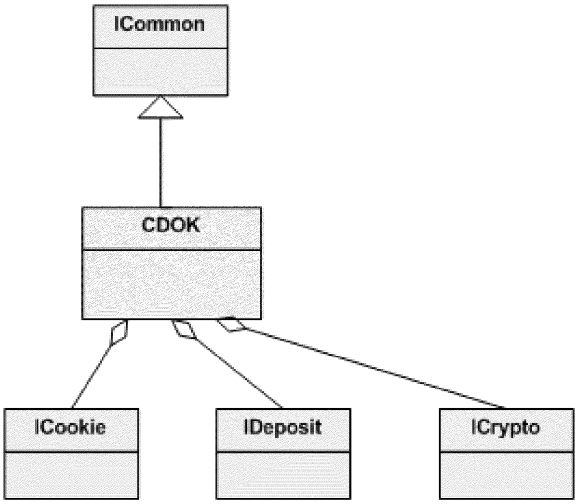


Figure 2-3: SDK Interfaces

The **CDOK** class inherits the **ICommon** interface, enabling the application to call all the **ICommon** interface methods directly. The **ICommon** interface methods provide the general operations that are likely to be performed on the DiskOnKey by all applications, for example, returning the DiskOnKey capacity or serial number.

The **CDOK** class aggregates the other DiskOnKey interfaces:

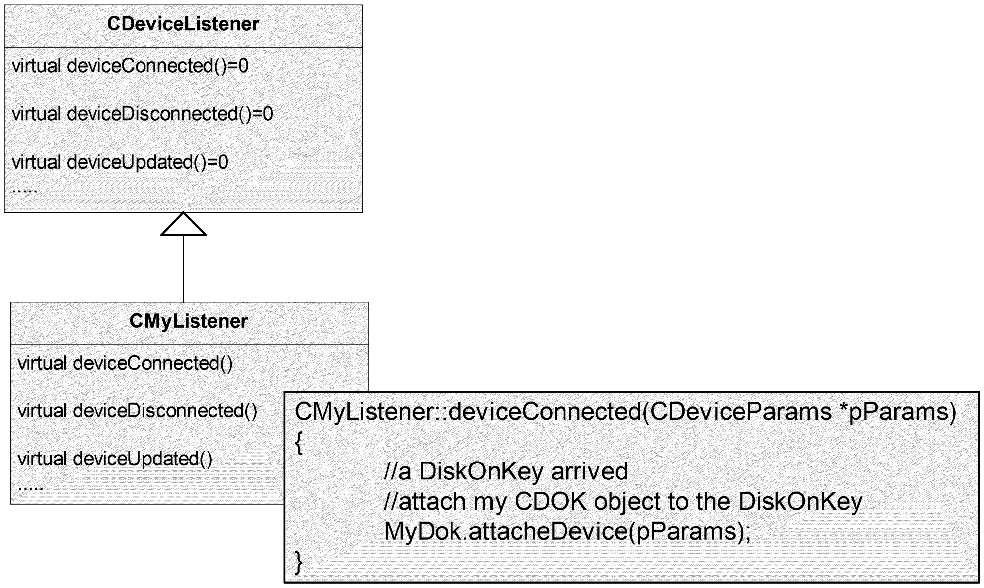
- The **ICookie** interface enables an application to manage a cookie on the DiskOnKey. The cookie resides in the hidden (reserved) area on the DiskOnKey and is used to save pieces of integer and string information generated by an application, ready for future access. Refer to Cookie API.
- The **IDeposit** interface enables the user application to create a password protected Safe area on the DiskOnKey, and to utilize the deposit capabilities. Refer to Deposit API .
- The **ICrypto** interface enables PKI based authentication. It enables the use of the DiskOnKey cryptography capabilities. Refer to Cryptography API .

2.3 Device Listening

Device listening is the mechanism for monitoring when a DiskOnKey is inserted or removed. Once the DiskOnKey device is connected, the application can call the **CDOK** methods.

2.3.1 Overview

The SDK forwards events dealing with device insertion or removal to a listening object, derived from the **CDeviceListener** interface. When the DiskOnKey is inserted, a **CDeviceListener** interface method attaches the **CDOK** object to a specific DiskOnKey device. The SDK provides methods that enable the **CDOK** object to work with that device, that is, run **CDOK** methods and APIs. Refer Device Listening Implementation Example. Figure 2-4 illustrates the device listening methods.


CDeviceListener
virtual deviceConnected()=0
virtual deviceDisconnected()=0
virtual deviceUpdated()=0
.....
CMyListener
virtual deviceConnected()
virtual deviceDisconnected()
virtual deviceUpdated()
.....

```
CMyListener::deviceConnected(CDeviceParams *pParams)
{
        //a DiskOnKey arrived
        //attach my CDOK object to the DiskOnKey
        MyDok.attacheDevice(pParams);
}
```

Figure 2-4: Device Listening Interfaces

2.3.2 Device Listening Methods

When the DiskOnKey is inserted, the **CDeviceListener** interface calls back the **CMyListener::deviceConnected** method and when the DiskOnKey is removed, the **CDeviceListener** interface calls back the **CMyListener::deviceDisconnected** method. See Figure 2-4: Device Listening Interfaces.

2.3.3 Device Listening Sequence

Figure 2-5 depicts the sequence of events that occurs in the device listening process and details the interaction that exists between the SDK Core and the **`CDeviceListener`** interface.

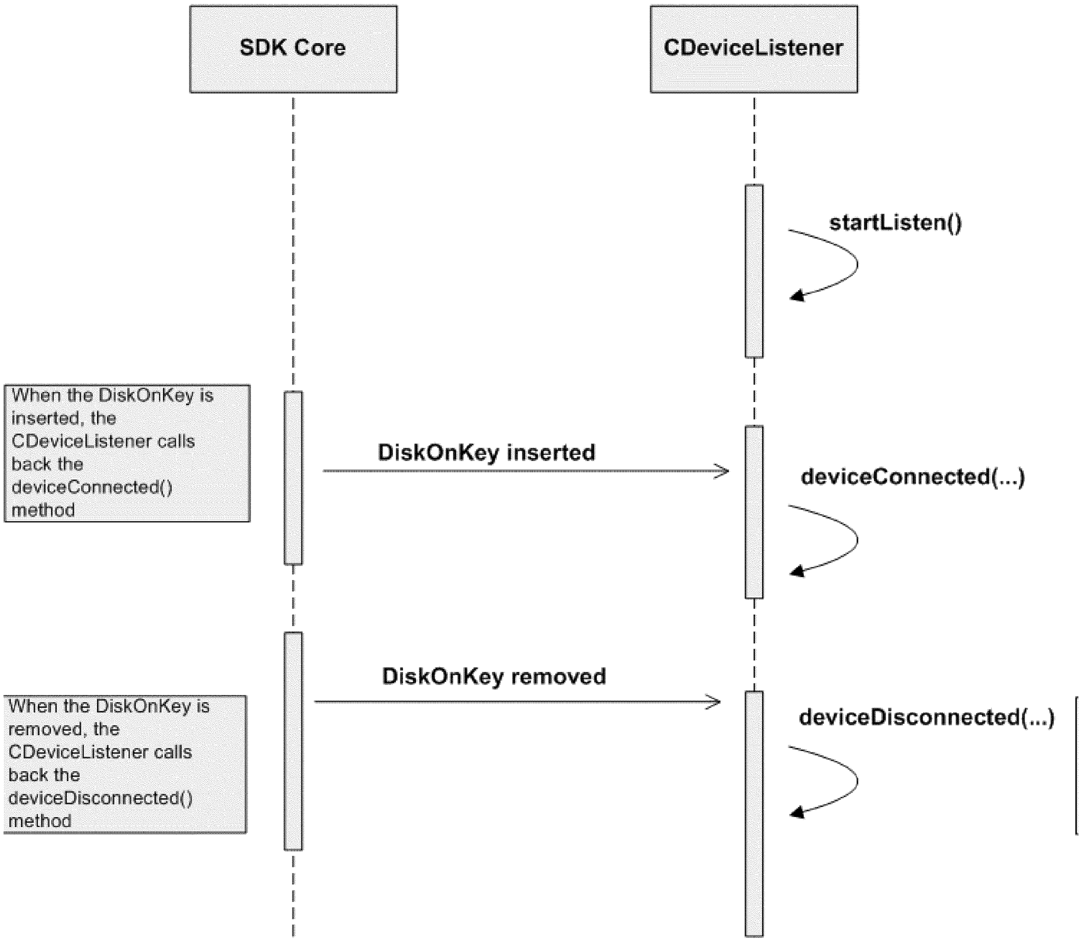

SDK Core
CDeviceListener
startListen()
When the DiskOnKey is inserted, the CDeviceListener calls back the deviceConnected() method
DiskOnKey inserted
deviceConnected(...)
DiskOnKey removed
When the DiskOnKey is removed, the CDeviceListener calls back the deviceDisconnected() method
deviceDisconnected(...)

Figure 2-5: Device Listening Sequence

The following sequence is executed:

1. Initialize SDK.

2. Start listening to devices.

3. Wait for device arrival and removal events.

2.4 DiskOnKey Structure

The DiskOnKey provides a secured file-system area (disk) that can be accessed via the SDK API. This area, called the Deposit partition, is a part of the DiskOnKey flash memory, and is hidden from the operating system. It enables the DiskOnKey user to utilize a secured area inside the DiskOnKey.

To expose the Deposit area, you must insert the deposit password, via the SDK API. The DiskOnKey authenticates the password, and then exposes the Deposit area to the operating system.

The SDK provides a set of API methods that enable the user application to manipulate the DiskOnKey deposit, for example to:

- Configure the deposit size
- Enter or exit the Deposit area
- Change the deposit password

The DiskOnKey comes in two configurations:

- A removable device
- A fixed device

In the removable DiskOnKey, there is only one partition displayed at any one time, either Public or Safe. When you enter the Safe area, the Safe area partition replaces the Public area partition.

In the fixed DiskOnKey, there are two partitions displayed at any one time, Public and Dummy. When you enter the Safe area, a Safe partition replaces the Dummy partition. During this replacement process, the fixed DiskOnKey disconnects from and reconnects to the operating system.

Note: In Windows 2000/Windows XP/Windows Me, the DiskOnKey operating system icons are different.

3 Building Applications for DiskOnKey

The DiskOnKey SDK currently runs on Win32 platforms; refer to Operating Systems Supported. The entire release is compiled using the Microsoft Visual C++ 6.0 compiler. The SDK is built as both a static (LIB) and dynamic (DLL) library, using the Win32 installation process described below.

3.1 Installing and Compiling

1. If you are upgrading from a previous release, use the **Clean** command in Microsoft Visual C++. You may have to review the application directories and delete all *.obj, *.dll, *.lib, *.ilk, *.pdb, *.idb, *.ncb, *.opt, and *.exp files, as the **Clean** command may not delete those files.
2. Uncompress the SDK distribution into a directory, in which it will create a **DOKSDK** directory containing the distribution. The **DOKSDK** directory will be referred to as **SDK_ROOT** in the following steps. For example, **SDK_ROOT\dok** would be **C:\DOKSDK\dok** if you were to uncompress into the root directory.

   Note: All the required libraries and include files are stored in the **SDK_ROOT\dok** directory. Ensure that you add the **SDK_ROOT** path to your additional include directories, and the **SDK_ROOT\dok** path to your additional libraries directory.

   Note: The library has been compiled with the **LIBCMT.LIB** run-time library. Any application linked to the SDK library should be compiled with this run-time library ('Multithreaded' C++ - Code Generation category).

   Note: If you use the dynamic libraries, make sure that you include **SDK_ROOT\bin** in the path whenever you run programs that use **DOKSDK**. Otherwise, you may experience problems finding **DOKSDK.dll**.

3.2 Naming Scheme

M-Systems, using Microsoft Visual C++, has used the following rules to name the **DLL** and **LIB** files in the DiskOnKey SDK:

(If using MFC? ("s" for static library | "d" for shared DLL) : "") + "Library/DLL name" + (If static library? "s" : "") + (If debugging enabled? "d" : "") + {".dll"|".lib"}

SDK Libraries

xDOKSDKx.lib, **xDOKUtilsx.lib**, **xRSAx.lib**, **xDOKRegHandlerx.lib**

SDK Samples

The samples are located in **SDK_ROOT\samples**. This directory includes a workspace, **samples.dsw**, which should be used to build the samples.

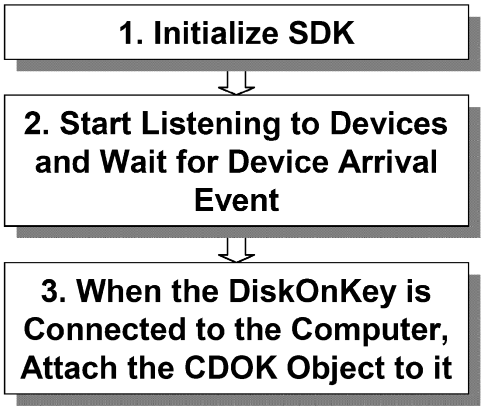
4 API Examples

The following sections present a number of API examples illustrating the varied capabilities of the SDK, including:

- Device Listening Implementation Example
- Simple Example
- Cookie Example
- Cryptography Example
- Deposit Example

4.1 Device Listening Implementation Example

The following workflow and code describes an example of listening to a DiskOnKey device.


1. Initialize SDK
2. Start Listening to Devices and Wait for Device Arrival Event
3. When the DiskOnKey is Connected to the Computer, Attach the CDOK Object to it

1. Initialize SDK

The **DOKSDK_init** and the **DOKSDK_fini** methods should be, respectively, the first and last methods invoked, when using the SDK. These methods are invoked only once, and their purpose is to initialize certain SDK objects.

In a Win32 application, **DOKSDK_init** may be invoked at the beginning of the **main()** procedure or in a static object, if the object uses the SDK.

In an MFC application, the **DOKSDK_init** method is invoked in **CWinApp::InitInstance**. The **DOKSDK_fini** method is invoked last, for example, in **CWinApp::ExitInstance**.

The event created in this example holds the main thread until the DiskOnKey events arrive.

```
{
DOKSDK_init();
hArrivalEvent = CreateEvent(NULL, false, false, NULL);
CMyListener listener;
```

2. Start Listening to Devices and Wait for Device Arrival Event

When calling the **startListen** method, the SDK scans all drive letters to search for a currently connected DiskOnKey. The SDK then waits for new device arrival or old device removal events from the operating system.

```
if(!listener.startListen())
return -1;
WaitForSingleObject(hArrivalEvent, INFINITE);
```

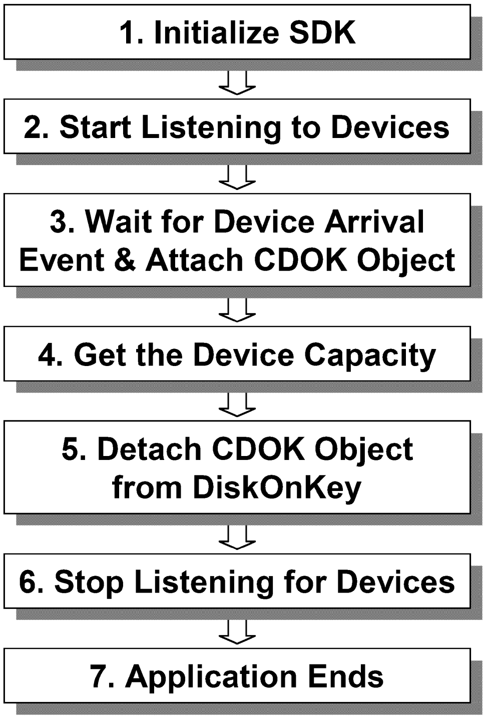
3. When the DiskOnKey is Connected to the Computer, Attach the CDOK Object to it

When detecting a DiskOnKey, the SDK calls back the pure virtual method **CDeviceListener::deviceConnected** that executes the user code for the new DiskOnKey.

```
void CMyListener::deviceConnected(const CDeviceParams* pParams)
{//a DiskOnKey arrived
            //attach the new DiskOnKey to the myDok object
            myDok.attachDevice(pParams);
//release the main
            SetEvent(hArrivalEvent);
}
```

4.2 Simple Example

The following workflow and code describes a possible scenario for working with the **`ICommon`** interface and illustrates how to use of the **`getDeviceCapacity`** function to retrieve the device capacity.


1. Initialize SDK
2. Start Listening to Devices
3. Wait for Device Arrival Event & Attach CDOK Object
4. Get the Device Capacity
5. Detach CDOK Object from DiskOnKey
6. Stop Listening for Devices
7. Application Ends

4. Get the Device Capacity

Call **`getDeviceCapacity`** from a `CDOK` object, after it is attached to a DiskOnKey device. The capacity returned from this method is the disk size of the DiskOnKey, not including the file-system space. The disk size of the DiskOnKey returned will be slightly different from the size returned by the operating system.

```
//get the device capacity
     opResult = myDok.getDeviceCapacity(ulDeviceCapacity);

     if(opResult == DOK_OK)
          printf("DiskOnKey capacity is %d bytes\n",
ulDeviceCapacity);
     else
          printf("error %s\n", CSDKUtils::errorString(opResult));
```

4.3 Cookie Example

The following code example describes a possible scenario for working with the **`ICookie`** API, how to write a string to/read from the hidden area, and how to delete an (ini) section.

1. Write a String to the Hidden Space

```
char szString[100], *pStrRes;
int nFreeHiddenSpace;
DOK_STATUS     eStatus;
sprintf(szString,"this is a test string");

if((eStatus = myDok.cookieAPI()->writeHiddenString("test", "tests
     string", szString))!= DOK_OK)
     {
          printf("error %s\n", CSDKUtils::errorString(eStatus));
          return -1;
     }
```

2. Read a String from the Hidden Space

If the allocated buffer is not big enough for the string size, the method returns a **`BUFFER_TOO_SMALL`** error. The required size, including the null termination, will appear in the string size argument.

```
     int nStrSize = 0;
     //get the necessary space for string allocation
     eStatus = myDok.cookieAPI()->readHiddenString("test", "tests
     string", NULL, nStrSize);
     if(eStatus == BUFFER_TOO_SMALL)
     {
          //allocate the buffer
          pStrRes = new char[nStrSize];
          //read the string and fill the new buffer
          eStatus = myDok.cookieAPI()->readHiddenString("test",
          "tests string", pStrRes, nStrSize);
}
          if(eStatus == DOK_OK)
     {
          printf("string result %s\n", pStrRes);
          delete[] pStrRes;
     }
     else
     {
          printf("error %s\n", CSDKUtils::errorString(eStatus));
          return -1;
     }
```

3. Get Free Space before Entry Deletion

If there is no space available for writing hidden data, the writing methods return a **`DOK_FULL`** error. Please note that this is only an illustration of how the hidden area space is captured by writing and freed by deleting. There is no need to call **`freeHiddenSpace`** before writing to the hidden area.

```
    if((eStatus = myDok.cookieAPI()-
    >freeHiddenSpace(nFreeHiddenSpace)) != DOK_OK)
    {
        printf("error %s\n", CSDKUtils::errorString(eStatus));
        return -1;
    }
    else
        printf("free hidden space %d\n", nFreeHiddenSpace);
```

4. Delete (ini) Section

Deleting a section deletes all data, entries and values contained within it.

```
    if((eStatus = myDok.cookieAPI()->delHiddenSection("test")) !=
    DOK_OK)
    {
        printf("error %s\n", CSDKUtils::errorString(eStatus));
        return -1;
    }
    else
        printf("deleted hidden section %s\n", "test");
```

5. Get Free Space after Entry Deletion

```
    if((eStatus = myDok.cookieAPI()-
    >freeHiddenSpace(nFreeHiddenSpace)) != DOK_OK)
    {
        printf("error %s\n", CSDKUtils::errorString(eStatus));
        return -1;
    }
    else
        printf("free hidden space %d\n", nFreeHiddenSpace);
```

4.4 Cryptography Example

The following code example describes a possible scenario for working with the **`Crypto`** API and illustrates how to encrypt and decrypt a buffer, as well as generate a digital signature of a file.

1. Build a Buffer to Encrypt

```
     BYTE encKey[24];

     BYTE bufferToEncrypt[1016];
     BYTE encryptedBuffer[1016];
     BYTE decryptedBuffer[1016];
     //build a buffer to encrypt
     for(int i=0;i<1016;i++)
          bufferToEncrypt[i] = (BYTE)i;
```

2. Produce a Random Encryption Key

This method produces a 24Byte random encryption key using the SDK **`makeRandom`** method.

```
     myDok.cryptoAPI()->makeRandom(encKey, 24);
```

3. Encrypt the Buffer

The method copies the encrypted buffer to the third argument passed to this method. The buffer will be encrypted inside the DiskOnKey using the DiskOnKey encryption algorithm.

```
     if((eError = myDok.cryptoAPI()->encryptBuffer(encKey,
bufferToEncrypt, encryptedBuffer, 1016)) != DOK_OK)
     {
          printf("error %s\n", CSDKUtils::errorString(eError));
          return -1;
     }
```

4. Decrypt the Buffer

```
     if((eError = myDok.cryptoAPI()->decryptBuffer(encKey,
encryptedBuffer, decryptedBuffer, 1016)) != DOK_OK)
     {
          printf("error %s\n", CSDKUtils::errorString(eError));
          return -1;
     }
```

5. Produce a Random Certificate

This method produces a random certificate for making a unique file signature.

```
     BYTE certificate[43];
     BYTE signature[128];//two signatures

     memset(signature, 0, 128);
     //produce a random certificate
     myDok.cryptoAPI()->makeRandom(certificate, 43);
```

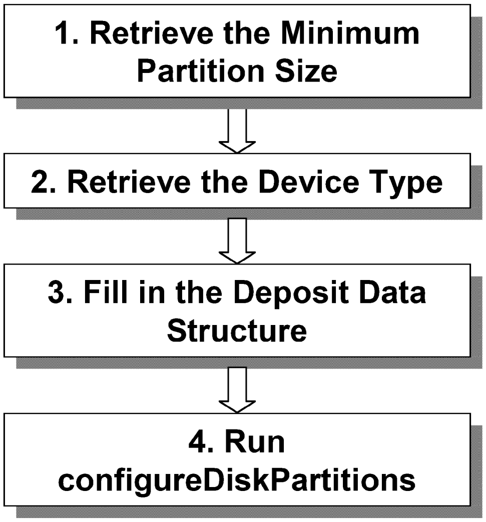
6. Sign the File

This method creates a digital signature of a file.

```
     if((eError = myDok.cryptoAPI()->digitalSignature("crypto.dsp",
certificate, signature)) != DOK_OK)
     {
          printf("error %s\n", CSDKUtils::errorString(eError));
          return -1;
     }
```

4.5 Deposit Example

The following workflow and code describes a possible scenario for working with the **`IDeposit`** API and illustrates how to configure a partition on the DiskOnKey.



1. Retrieve the Minimum Partition Size

2. Retrieve the Device Type

3. Fill in the Deposit Data Structure

4. Run configureDiskPartitions

1. Retrieve the Minimum Partition Size

Any attempt to create a smaller partition will return an error.

```
unsigned long ulSafeSize, ulDummySize;
     CDeviceParams::_deviceType deviceType;
     DOK_STATUS eError;
          if(myDok.depositAPI()->minPartitionSize(ulSafeSize,
ulDummySize) != DOK_OK)
          return -1;
```

2. Retrieve the Device Type

```
if(myDok.getDeviceType(deviceType) != DOK_OK)
          return -1;
```

3. Fill in the Deposit Data Structure

```
DepositConfig depConf;
strcpy(depConf.szPaassword,"My_Password");//Safe partition password
     strcpy(depConf.szHint,"My_Hint");//Safe partition password
hint
     if(deviceType == CDeviceParams::_deviceType::DP_FIXED_DISK)
     {//fill fixed disk parameters
          depConf.fp.nDummyPartitionByteSize = ulDummySize;
          depConf.fp.nSafePartitionByteSize = ulSafeSize;
          strcpy(depConf.fp.szDummyPartitionLabel, "My Dummy");
          strcpy(depConf.fp.szPublicPartitionLabel,"My Public");
          strcpy(depConf.fp.szSafePartitionLabel, "My Safe");
     }
     else
     {//removable disk parameters
          depConf.rp.nSafePartitionByteSize = ulSafeSize;
          strcpy(depConf.rp.szPublicPartitionLabel, "My Public");
          strcpy(depConf.rp.szSafePartitionLabel, "My Safe");
     }
```

4. Run configureDiskPartitions

This action reformats the DiskOnKey and erases all data in its file system (not in the cookie area).

```
opHandler callBack;
if((eError = myDok.depositAPI()->configureDiskPartitions(depConf,
&callBack, true)) != DOK_OK)
     {
          printf("error %s\n", CSDKUtils::errorString(eError));
          return -1;
     }
     else
          printf("configure disk partition success\n");
```

5 SDK APIs

Most of the methods return **DOK_STATUS**, an enumerator, whose values may be found in the **DeviceParams.h** file. For detailed error information, refer to Error Codes. If the method succeeds, the status returned will be **DOK_OK**. The actual return value(s) of a method will be contained in the parameters passed by reference to the method.

The developer is responsible for allocating the buffer for char buffers. It is therefore possible that the allocated buffer size will be too small. In the event of this happening, a **BUFFER_TOO_SMALL** error is returned, and the required buffer size is specified in one of the method's parameters.

5.1 Quick Reference

Table 5-1 may be used as a quick reference tool, linking to the API interfaces and cross-referencing the relevant page.

Table 5-1: API Quick Reference

API Interface	Page
CDOK Class Methods	28
Cookie API	47
Deposit API	58
Cryptography API	71

Table 5-2 shows the interfaces supported by each firmware version.

Table 5-2: Firmware Versions

DiskOnKey FW Version	ICommon	ICookie	IDeposit	ICrypto
2.01	Yes	Yes	No	No
2.51	Yes	Yes	Yes	Partial
3.x	Yes	Yes	Yes	Yes

5.2 API Access Points

The ***xxx*API()** methods serve as the access points via which the user application exposes the specific APIs exported by the **CDOK** object. These methods return a pointer to a specific API abstract class.

These APIs enable an application to utilize all the additional functionality that the DiskOnKey firmware provides.

Table 5-3 provides a list of the access points and links to the specific APIs.

Table 5-3: API Methods

Method	Description	Page
cryptoAPI()	Access point to the Cryptography API	71
depositAPI ()	Access point to the Deposit API	58
cookieAPI()	Access point to the Cookie API	47

5.3 IDOKOpHandler CallBack

The **`IDOKOpHandler`** callback interface is passed to methods that can run synchronously or asynchronously. The progress and result of these methods will call back the appropriate method in the **`IDOKOpHandler`** callback derived object. The behavior of the callback will be the same whether running the methods synchronously or asynchronously. Passing this interface to the methods is not mandatory.

A *`bBlock`* parameter specifies when the method will be returned to the user. If *`bBlock`* is true, the method will run synchronously and the method will return to the user when the DiskOnKey is reconnected. If *`bBlock`* is false, the method will run asynchronously and will return to the user before the DiskOnKey is disconnected. A different thread is opened for the disconnect, which may result in an error being received from the callback.

5.4 CDeviceListener Object

The programmer overwrites specific methods to enable the application to be constantly updated regarding the DiskOnKey connection status.

After initializing the DOKSDK, you must create an instance of **`CMyListener`** and start listening to devices by running the **`startlisten`** method. When you run this function, the SDK opens a second parallel thread, on which the SDK checks for devices to be connected.

The SDK first checks that a DiskOnKey is present. Then it runs an event loop waiting for events from the operating system. The events may include changes in device status, such as connected, disconnected and arrived. Before the SDK thread is issued, you must check that the device is not already connected at the time that the SDK thread is created. If the device is already connected, the SDK thread will not receive events emanating from Windows.

When the device arrives, the SDK thread receives an event from Windows. The device is then authenticated to verify that it is a DiskOnKey. If it is a DiskOnKey, the SDK passes the user application callback to notify the user application that the device is connected. The SDK thread does not synchronize between the two threads. The user application overrides the device connect and is responsible for synchronizing between the two threads.

Table 5-4 describes the Device Listening methods.

Table 5-4: Device Listening Methods

Method	Description
startListen	After initializing the SDK, the **`startListen`** method is invoked.
stopListen	After the CDOK object is detached and before the application ends, the **`stopListen`** method is invoked.
Callback Methods	
deviceConnected	When a DiskOnKey arrived event is received, the device is connected.
deviceDisconnected	When a DiskOnKey removed event is received, the device is disconnected.
deviceUpdated	If the device is modified, for example, if the device has been formatted and now contains another partition (fixed disk configuration), the **`CDeviceParams`** drive letter mask is updated. If a **`CDOK`** object is attached to this device, the mask is updated automatically
unknownDeviceConnected	When a device arrives and the SDK does not recognize its strings, the SDK will interrogate the device. If it is a DiskOnKey, the SDK calls the **`CMyListener::deviceConnected method`**. If the **`unknownDeviceConnected`** method returns false, the SDK ignores the device. (Relevant for firmware version < 3.0.)

5.5 CDOK Class Methods

The **CDOK** class inherits the **ICommon** interface, enabling the application to call all the **ICommon** interface methods directly.

The **ICommon** interface methods provide the general operations that are likely to be performed on the DiskOnKey by all applications, for example, returning the DiskOnKey capacity or serial number.

Table 5-5 lists the **CDOK** class methods.

Table 5-5: CDOK Class Methods

Method	Description	Page
attachDevice	Attaches the CDOK object to a DiskOnKey by its drive letter.	29
attachDevice	Attaches the CDOK object to a DiskOnKey by passing the device parameters.	30
detachDevice	Detaches the CDOK object from the DiskOnKey.	31
isDeviceOK	Checks if the DiskOnKey is functional.	32
sdkVersion	Returns the SDK version.	33
getDeviceCapacity	Returns the DiskOnKey capacity in bytes.	34
getDeviceSerialNumber	Returns the DiskOnKey serial number.	35
getDeviceVersion	Returns the DiskOnKey firmware version.	36
canAccessDevice	Checks if the SDK can access disk devices. For example, when the SDK is not running with administrator privileges.	37
ejectDevice	Stops/ejects the DiskOnKey device.	38
getDriveStrings	Returns the DiskOnKey drive letters string.	39
partitionType	Returns the type of a DiskOnKey partition.	40
getDeviceType	Returns the DiskOnKey type, that is, removable or fixed.	41
isWriteProtected	Verifies if the DiskOnKey is write-protected.	42
updateWriteProtected	Updates the write protection status of the DiskOnKey.	43
writeImage	Writes an image file to the DiskOnKey.	45
format	Formats the DiskOnKey.	46
Cookie API	Access point to the CDOK Cookie API.	47
Deposit API	Access point to the CDOK Deposit API.	58
Cryptography API	Access point to the CDOK Cryptography API.	71

5.5.1 attachDevice

Description

Attaches the CDOK object to a DiskOnKey by its drive letter.

Syntax

```
bool attachDevice(char cDriveLetter);
```

Parameters

cDriveLetter

[in] Specifies the DiskOnKey drive letter.

Return Values

If the function succeeds, the return value is **true**.

If the function fails, the return value is **false**.

For detailed error information, refer to Error Codes.

Remarks

1. This method will attach the **CDOK** object to a specific DiskOnKey device, and will enable the **CDOK** object to function with that device, that is, run CDOK methods and APIs.
2. The SDK automatically detaches the **CDOK** object from the DiskOnKey if the device is ejected (or if the device is removed). The **CDOK** object is not automatically re-attached if the DiskOnKey is plugged in.
3. The SDK will automatically change the CDOK status to 'not connected'. Every additional function call will return a **DOK_NOT_CONNECTED** status.
4. In order to re-enable the **CDOK** object to function with the DiskOnKey device, re-attach the **CDOK** object.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **DOK.h**.

See Also:

attachDevice, detachDevice, isDeviceOK, CDOK Object, CDOK Class Methods

5.5.2 attachDevice

Description

Attaches the **CDOK** object to a DiskOnKey by passing the device parameters.

Syntax

```
bool attachDevice (const CDeviceParams* pDeviceParams);
```

Parameters

pDeviceParams

[in] Specifies the DiskOnKey device parameters.

Return Values

If the function succeeds, the return value is **true**.

If the function fails, the return value is **false**.

For detailed error information, refer to Error Codes.

Remarks

Same as in attachDevice.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **DOK.h**.

See Also:

attachDevice, detachDevice, isDeviceOK, CDOK Object, CDOK Class Methods

5.5.3 detachDevice

Description

Detaches the **CDOK** object from the DiskOnKey.

Syntax

```
void detachDevice();
```

Parameters

None.

Return Values

None.

Remarks

1. Detach the **CDOK** object from the DiskOnKey device to which it is attached.
2. After detaching the **CDOK** object, any method call will return a **DOK_NOT_CONNECTED** status.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **DOK.h**.

See Also:

attachDevice, attachDevice, detachDevice, isDeviceOK, CDOK Object, CDOK Class Methods

5.5.4 isDeviceOK

Description

Checks if the DiskOnKey is functional.

Syntax

```
bool isDeviceOK();
```

Parameters

None.

Return Values

If the status of the DiskOnKey is OK, this method will return true meaning that the device can be addressed by the SDK.

If the device cannot be addressed by the SDK, this method will return false.

For detailed error information, refer to Error Codes.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in `DOK.h`.

See Also:

attachDevice, attachDevice, detachDevice, CDOK Object, CDOK Class Methods

5.5.5 sdkVersion

Description

Returns the SDK version.

Syntax

```
const char* sdkVersion;
```

Parameters

None.

Return Values

This method returns the SDK version string.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **`DOK.h`**.

See Also:

getDeviceSerialNumber, getDeviceId, getDeviceVersion, getDeviceType, CDOK Object, CDOK Class Methods

5.5.6 getDeviceCapacity

Description

Returns the DiskOnKey capacity in bytes.

Syntax

```
DOK_STATUS getDeviceCapacity(unsigned long& ulByteSize);
```

Parameters

ulByteSize

[out] Receives the DiskOnKey capacity in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK** and *ulByteSize* contains the DiskOnKey capacity, that is, the byte size of the DiskOnKey.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The DiskOnKey capacity ranges between 8 and 512MB, expressed in bytes.
2. The DiskOnKey capacity value returned by this function may not be identical to the value displayed in Windows Explorer, as a result of file system space allocation.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

getDeviceSerialNumber, getDeviceId, getDeviceVersion, getDeviceType, CDOK Object, CDOK Class Methods

5.5.7 getDeviceSerialNumber

Description

Returns the DiskOnKey serial number.

Syntax

```
DOK_STATUS getDeviceSerialNumber(char* szSerial, int &nStrLength);
```

Parameters

szSerial

[out] Receives the DiskOnKey serial number.

nStrLength

[in/out] Specifies the size of the *szSerial* buffer.

Return Values

If the function succeeds, the return value is **DOK_OK** and *szSerial* contains the DiskOnKey serial number string.

If the function fails, the return value is != **DOK_OK**.

If the buffer pointed to by *szSerial* is not large enough, a **BUFFER_TOO_SMALL** error is returned, and *nStrLength* specifies the required buffer size.

For detailed error information, refer to Error Codes.

Remarks

The DiskOnKey serial number is different for each device. It is a unique identifier for a DiskOnKey device.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

getDeviceCapacity, getDeviceId, getDeviceVersion, getDeviceType, CDOK Object, CDOK Class Methods

5.5.8 getDeviceVersion

Description

Returns the DiskOnKey firmware version.

Syntax

```
DOK_STATUS getDeviceVersion(char* szVersion, int &nStrLength );
```

Parameters

szVersion

[out] Receives the DiskOnKey firmware version.

nStrLength

[in/out] Specifies the required buffer size for *szVersion*. *nStrLength* should be the size of *szVersion*.

Return Values

If the function succeeds, the return value is **DOK_OK** and *szVersion* contains the DiskOnKey version.

If the function fails, the return value is != **DOK_OK**.

If the buffer pointed to by *szVersion* is not large enough, a **BUFFER_TOO_SMALL** error is returned, and *nStrLength* specifies the required buffer size.

For detailed error information, refer to Error Codes.

Remarks

1. The syntax of the version is *x.xx*.
2. The following table shows the interfaces supported by each version:

DiskOnKey FW Version	ICommon	ICookie	IDeposit	ICrypto
2.01	Yes	Yes	No	No
2.51	Yes	Yes	Yes	Partial
3.x	Yes	Yes	Yes	Yes

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

getDeviceCapacity, getDeviceSerialNumber, getDeviceId, getDeviceType, CDOK Object, CDOK Class Methods

5.5.9 canAccessDevice

Description

Checks if the SDK can access disk devices.

Syntax

```
DOK_STATUS canAccessDevice();
```

Return Values

If the SDK can access a disk device, the return value is **DOK_OK**.

If the function fails, an **ACCESS_ERR** error is returned.

For detailed error information, refer to Error Codes.

Remarks

In Windows XP and Windows 2000 the user must be logged in as an Administrator. If not, the SDK cannot access the device. Therefore, it is crucial to verify access for these operating systems. This issue will be resolved in a future SDK version.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

Operating Systems: Windows 2000, Windows XP

See Also:

CDOK Object, CDOK Class Methods

5.5.10 ejectDevice

Description

Stops/ejects the DiskOnKey device by disconnecting the USB data lines to the DiskOnKey. The DiskOnKey is no longer available as a disk drive, and the drive is no longer visible in Windows Explorer.

Syntax

```
DOK_STATUS ejectDevice(bool &bResult);
```

Parameters

bResult

[out] Specifies whether or not the DiskOnKey was ejected.

Return Values

If the function succeeds, the return value is **DOK_OK** and *bResult* is true indicating that the DiskOnKey was successfully stopped/ejected. If *bResult* is false, the DiskOnKey was not stopped/ejected.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. The DiskOnKey will not be ejected if an application or file has an open handle to the device. For example, if a file stored on the DiskOnKey is open.
2. When copying large files, Windows XP and Windows 2000 performs a lazy write, buffering the data first before writing it to the hard disk. Although it may appear to the user that the copy operation is complete, if the DiskOnKey is unplugged without safely ejecting, data may be lost.
3. The SDK automatically detaches the **CDOK** object from the DiskOnKey if the device is ejected (or if the device is removed). The **CDOK** object is not automatically re-attached if the DiskOnKey is plugged in.
4. Once the DiskOnKey has been ejected, it must be unplugged and then plugged in again before it becomes available again.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

Operating Systems: Windows 2000, Windows XP

See Also:

CDOK Object, CDOK Class Methods

5.5.11 getDriveStrings

Description

Returns the DiskOnKey drive letters string. This includes all the drive letters that make up this DiskOnKey, for example, all partitions on a fixed DiskOnKey. In addition, the Public area and the Safe area (commonly referred to as the Privacy Zone) on a fixed DiskOnKey will each appear as a drive letter. The removable DiskOnKey appears as one drive only.

Syntax

```
DOK_STATUS getDriveStrings(char* pBuffer, int &nBufferLength);
```

Parameters

pBuffer

[out] Pointer to a buffer that receives the DiskOnKey drive letter strings.

nBufferLength

[in/out] Specifies the required buffer size for *pBuffer*. The size includes the terminating null character.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pBuffer* contains the DiskOnKey drive letter strings.

If the function fails, the return value is != **DOK_OK**.

If the buffer pointed to by *pBuffer* is not large enough, a **BUFFER_TOO_SMALL** error is returned, and *nBufferLength* specifies the required buffer size.

For detailed error information, refer to Error Codes.

Remarks

1. The drive letter strings syntax is comprised of a series of null-terminated strings, one for each valid drive in the DiskOnKey. An additional null character ends the syntax, as illustrated below: “F:\< null>G:\< null>H:\< null>< null>”.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

partitionType, CDOK Object, CDOK Class Methods

5.5.12 partitionType

Description

Returns the type of a DiskOnKey partition.

Syntax

```
DOK_STATUS partitionType(const char* szPartition,
ICommon::ePartitionType &eType);
```

Parameters

szPartition

[in] Specifies the DiskOnKey partition drive.

eType

[out] Receives the DiskOnKey partition type.

Return Values

If the function succeeds, the return value is **DOK_OK** and *eType* contains the DiskOnKey partition type.

If the function fails, the return value is != **DOK_OK** and a **DRIVE_LETTER_ERR** error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. *szPartition* should be one of the DiskOnKey partition drives, for example "F".
2. *eType* values may be:
    - **PT_PUBLIC**, the Public partition.
    - **PT_SAFE**, the Safe partition.
    - **PT_DUMMY**, the Dummy partition, in the fixed device.
3. If the partition is not a DiskOnKey partition, or if the SDK can't recognize it, the **DRIVE_LETTER_ERR** is returned.
4. On a fixed disk, after formatting, the SDK writes **.PUBLIC**, **.DUMMY** and **.SAFE** files to their respective partitions. These are hidden files and cannot be deleted by the user using Windows 2000, Windows XP or Windows Me. In Windows 98 and Mac, the user can delete the files. These hidden files assist the **partitionType** function to identify the partition types. If these hidden files are deleted, for example, if the user formats the DiskOnKey, the **partitionType** function will not succeed.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

getDriveStrings, CDOK Object, CDOK Class Methods

5.5.13 getDeviceType

Description

Returns the DiskOnKey type, that is, removable or fixed.

Syntax

```
DOK_STATUS getDeviceType(CDeviceParams::eDeviceType &eType);
```

Parameters

eType

[out] Receives the DiskOnKey type.

Return Values

If the function succeeds, the return value is **DOK_OK** and *eType* contains the DiskOnKey type:

- **DP FIXED DISK**, for a fixed DiskOnKey.
- **DP REMOVABLE DISK,** for a removable DiskOnKey.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

CDOK Object, CDOK Class Methods

5.5.14 isWriteProtected

Description

Verifies if the DiskOnKey is write-protected or not.

Syntax

```
DOK_STATUS isWriteProtected(bool &bWriteProtect);
```

Parameters

bWriteProtect

[out] Specifies if the DiskOnKey status is write-protected or not.

Return Values

If the function succeeds, that is the return value is **DOK_OK**, and *bWriteProtect* is true, the device is write-protected. If *bWriteProtect* is false, the device is not write-protected.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

If you attempt to invoke this function on a DiskOnKey with a firmware version < 3.0, a **CMD_NOT_SUPPORTED** error is returned.

Requirements

DiskOnKey FW Version: Requires 3.0 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

updateWriteProtected, CDOK Object, CDOK Class Methods

5.5.15 updateWriteProtected

Description

Updates the write protection status of the DiskOnKey.

Syntax

```
DOK_STATUS updateWriteProtected(bool bWriteProtect, IDOKOpHandler*
pCallback, bool bBlock);
```

Parameters

bWriteProtect

[in] Specifies whether or not to write protect the device. If false, unprotect the device. If true, write protect the device.

pCallback

[out] The callback receives the method results.

bBlock

[in] Specifies if the method will block for DiskOnKey reconnection. If *bBlock* is true, the method will run synchronously and will block until the DiskOnKey will reconnect to the OS. If *bBlock* is false, the method will run asynchronously and won't block. The result will be passed by the callback API.

Return Values

If the function succeeds, the return value is **DOK_OK**, *pCallback* specifies whether the DiskOnKey reconnection operation succeeded or failed.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. The **updateWriteProtected** action requires DiskOnKey to be reconnected to the system, in order to notify the operating system of its new state. The SDK performs the reconnection automatically and blocks the events from reaching the SDK user. The **updateWriteProtected** function can be executed either synchronously or asynchronously (recommended for Win32 non-console applications), depending on the *bBlock* value.
2. If *bBlock* is true, the method will run synchronously and return to the user after the DiskOnKey is reconnected. If *bBlock* is false, the method will run asynchronously and return to the user before the DiskOnKey is disconnected. A different thread is opened for the disconnect, which may result in an error being received from the callback.

3. The **`IDOKOpHandler`** callback interface is passed to methods that can run synchronously or asynchronously. The progress and result of these methods calls back the appropriate method in the **`IDOKOpHandler`** callback derived object. The behavior of the callback will be the same whether using the methods, running synchronously or asynchronously. Passing this interface to the methods is not mandatory.
4. If you attempt to invoke this function in versions < 3.0, a **`CMD_NOT_SUPPORTED`** error is returned.

Requirements

DiskOnKey FW Version: Requires 3.0 or later.

Header: Declared in **`ICommon.h`**; included in **`DOK.h`**.

See Also:

isWriteProtected, CDOK Object, CDOK Class Methods

5.5.16 writeImage

Description

Writes an image file to the DiskOnKey.

Syntax

```
DOK_STATUS writeImage(const char* szFileName, int nStartSector = 0);
```

Parameters

szFileName

[in] Specifies the file absolute path to be written to the DiskOnKey.

nStartSector

[in] Specifies the starting sector of the DiskOnKey, to which the file is to be written.

Return Values

If the function succeeds, the return value is **DOK_OK** and the image has been successfully written to the DiskOnKey.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

This function should be used with great caution, as you can overwrite the DiskOnKey file system.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

CDOK Object, CDOK Class Methods

5.5.17 Format

Description

Formats the DiskOnKey.

Syntax

```
DOK_STATUS format(FormatConfig* formatConf);
```

Parameters

formatConf

[in] Specifies the format configuration structure, detailed in **DOKSDK_DEF.h**. Refer to Data Types.

Return Values

If the function succeeds, the return value is **DOK_OK** and the DiskOnKey has been formatted.

If the function fails, the return value is != **DOK_OK**, a **FORMAT_FAIL** error is returned.

For detailed error information, refer to Error Codes.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICommon.h**; included in **DOK.h**.

See Also:

CDOK Object, CDOK Class Methods

5.6 Cookie API

The **ICookie** interface enables an application to manage a cookie on the DiskOnKey. The cookie resides in the hidden (reserved) area on the DiskOnKey and is used to save pieces of integer and string information generated by an application, ready for future access. For example, the cookie can be used to save a password hint on the DiskOnKey. The cookie and hidden area are not directly accessible by the user.

Another example might be that the DiskOnKey is attached to a specific icon and label, based on the information stored in the cookie.

The DiskOnKey hidden area size may vary between 7K and 15K, depending on the DiskOnKey version:

- Version 2.x hidden area size is 7K.
- Version 3.x hidden area size is 15K.

The **ICookie** interface uses an ini-based API, similar to the Win32 ini API. The ini structure, as detailed in **ICookie.h**, is:

```
[section 1]
entry1 = value1
entry2 = value2
...
[section 2]
entry10 = value10
entry11 = value11
...
```

The entry names must be unique within a section.

Table 5-6: Cookie API

Method	Description	Page
writeHiddenString	Writes a string value to a specific section and entry in the DiskOnKey cookie.	49
writeHiddenInt	Writes an integer value to a specific section and entry in the DiskOnKey cookie.	50
readHiddenString	Extracts a string value contained in a specific section and entry in the DiskOnKey cookie.	51
readHiddenInt	Returns the integer value contained in a specific section and entry in the DiskOnKey cookie.	52
delHiddenEntry	Deletes an entry in the DiskOnKey cookie.	53
delHiddenSection	Deletes a section in the DiskOnKey cookie.	54
freeHiddenSpace	Returns the amount of available hidden space.	55
dumpHiddenArea	Copies the DiskOnKey cookie into a text file.	56
clearHiddenArea	Deletes the DiskOnKey cookie.	57
CDOK Class Methods	CDOK Class Methods API.	28
Deposit API	CDOK Deposit API.	58
Cryptography API	CDOK Cryptography API.	71

5.6.1 writeHiddenString

Description

Writes a string value to a specific section and entry in the DiskOnKey cookie.

Syntax

```
DOK_STATUS writeHiddenString(const char* szSection const char*
szEntry const char* szValue);
```

Parameters

szSection

[in] Specifies the DiskOnKey cookie section.

szEntry

[in] Specifies the entry in the section.

szValue

[in] Specifies the value to be written to the specified entry.

Return Values

If the function succeeds, the return value is **DOK_OK** and the string has been successfully written to the DiskOnKey hidden area.

If the function fails, the return value is != **DOK_OK**.

If you enter a null section or null entry, **PARAMS_ERR** is returned.

If there is no space left in the hidden area, **DOK_FULL** is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This function should be used with caution in order to prevent accidentally overwriting an existing value.
2. The section and entry length should not exceed 31 bytes.
3. The maximum length for the string value is 128 bytes.
4. Before performing the action, the SDK will extract all cookie data from the DiskOnKey, and then will update it if necessary. This method is thread/process safe.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

writeHiddenInt, readHiddenString, delHiddenEntry, dumpHiddenArea, CDOK Object, Cookie API

5.6.2 writeHiddenInt

Description

Writes an integer value to a specific section and entry in the DiskOnKey cookie.

Syntax

```
DOK_STATUS writeHiddenInt(const char* szSection const char* szEntry
int nValue);
```

Parameters

szSection

[in] Specifies the DiskOnKey cookie section.

szEntry

[in] Specifies the entry in the section.

nValue

[in] Specifies the integer value to be written to the specified entry.

Return Values

If the function succeeds, the return value is **DOK_OK** and the entry has been successfully written to the DiskOnKey hidden area.

If the function fails, the return value is != **DOK_OK**.

If you enter a null section or null entry, **PARAMS_ERR** is returned.

If there is no space left in the hidden area, **DOK_FULL** is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method will overwrite an existing value in the same section or entry and should therefore be used with caution.
2. The section and entry length should not exceed 31 bytes.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

writeHiddenString, readHiddenInt, delHiddenEntry, dumpHiddenArea, CDOK Object, Cookie API

5.6.3 readHiddenString

Description

Extracts a string value contained in a specific section and entry in the DiskOnKey cookie.

Syntax

```
DOK_STATUS readHiddenString(const char* szSection const char* szEntry
char* szResult int &nResultByteSize);
```

Parameters

szSection

[in] Specifies the DiskOnKey cookie section.

szEntry

[in] Specifies the entry in the section.

szResult

[out] Receives the string value contained in the specified section and entry in the DiskOnKey cookie.

nResultByteSize

[in/out] Specifies the size of *szResult*.

Return Values

If the function succeeds, the return value is **DOK_OK** and *szResult* contains the string value stored in the specific section and entry in the DiskOnKey hidden area.

If the function fails, the return value is != **DOK_OK**.

If the buffer pointed to by *szResult* is not large enough, a **BUFFER_TOO_SMALL** error is returned, and *nResultByteSize* specifies the required buffer size.

If the entry or section does not exist, a **DATA_NOT_FOUND** error will be returned.

If you enter a null section or null entry, a **PARAMS_ERR** is returned.

For detailed error information, refer to Error Codes.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

readHiddenInt, writeHiddenString, dumpHiddenArea, CDOK Object, Cookie API

5.6.4 readHiddenInt

Description

Returns the integer value contained in a specific section and entry in the DiskOnKey cookie.

Syntax

```
DOK_STATUS readHiddenInt(const char* szSection const char* szEntry
int &nResult);
```

Parameters

szSection

[in] Specifies the DiskOnKey cookie section.

szEntry

[in] Specifies the entry in the section.

nResult

[in] Receives the integer value contained in the specific section and entry in the DiskOnKey cookie.

Return Values

If the function succeeds, the return value is **DOK_OK** and *nResult* contains the integer value stored in the specific section and entry in the DiskOnKey hidden area.

If the function fails, the return value is != **DOK_OK**.

If the entry or section does not exist, a **DATA_NOT_FOUND** error will be returned.

If you enter a null section or null entry, a **PARAMS_ERR** is returned.

For detailed error information, refer to Error Codes.

Remarks

None.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

readHiddenString, writeHiddenInt, dumpHiddenArea, CDOK Object, Cookie API

5.6.5 delHiddenEntry

Description

Deletes an entry from the DiskOnKey cookie.

Syntax

```
DOK_STATUS delHiddenEntry(const char* szSection const char* szEntry);
```

Parameters

szSection

[in] Specifies the DiskOnKey hidden area section.

szEntry

[in] Specifies the entry in the section.

Return Values

If the function succeeds, the return value is **DOK_OK** and the entry is deleted.

If the function fails, the return value is != **DOK_OK**.

If the entry or section does not exist, a **DATA_NOT_FOUND** error will be returned.

If you enter a null section or null entry, a **PARAMS_ERR** is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method will delete an entry and the value it contains from the hidden area.
2. There is no way of undoing the method action.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

writeHiddenString, writeHiddenInt, delHiddenSection, dellAllCookies, dumpHiddenArea, CDOK Object, Cookie API

5.6.6 delHiddenSection

Description

Deletes a hidden section.

Syntax

```
DOK_STATUS delHiddenSection(const char* szSection);
```

Parameters

szSection

[in] Specifies the DiskOnKey hidden area section.

Return Values

If the function succeeds, the return value is **DOK_OK** and the section is deleted.

If the function fails, the return value is != **DOK_OK.**

If the section does not exist, a **DATA_NOT_FOUND** error will be returned.

If you enter a null section, a **PARAMS_ERR** is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method will delete a section, and all the entries and values under it, from the hidden area.
2. There is no way of undoing the method action.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

delHiddenEntry, dellAllCookies, dumpHiddenArea, CDOK Object, Cookie API

5.6.7 freeHiddenSpace

Description

Returns the amount of available hidden space.

Syntax

```
DOK_STATUS freeHiddenSpace(int &nFreeByteSpace);
```

Parameters

nFreeByteSpace

[out] Returns the amount of available hidden space.

Return Values

If the function succeeds, the return value is **DOK_OK** and *nFreeByteSpace* contains the amount of free hidden space currently available.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

There is a space overhead for every entry. This means that the space occupied after calling a cookie method, such as, **writeHiddenInt**, is larger then the size of the integer and entry.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

dellAllCookies, dumpHiddenArea, CDOK Object, Cookie API

5.6.8 dumpHiddenArea

Description

Dumps the hidden area data into a text file.

Syntax

```
DOK_STATUS dumpHiddenArea(const char* szFileName);
```

Parameters

szFileName

[in] Specifies the path of a text file into which the hidden area data is to be dumped.

Return Values

If the function succeeds, the return value is **DOK_OK** and the cookie data was written to the text file.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The method dumps the hidden area to a text file.
2. The file syntax resembles an ini file syntax.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

dellAllCookies, CDOK Object, Cookie API

5.6.9 clearHiddenArea

Description

Deletes the cookie.

Syntax

```
DOK_STATUS clearHiddenArea ();
```

Parameters

None.

Return Values

If the function succeeds, the return value is **DOK_OK** and the cookie is deleted.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. This method clears the hidden area and deletes all data.
2. The method action cannot be undone.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICookie.h**; included in **DOK.h**.

See Also:

dumpHiddenArea, CDOK Object, Cookie API

5.7 Deposit API

The **IDeposit** interface enables the user application to create a Safe area on the DiskOnKey. In the removable DiskOnKey, there is only one partition displayed at any one time, either Public or Safe. When you enter the Safe area, the Safe area partition replaces the Public area partition.

In the fixed DiskOnKey, there are two partitions displayed at any one time, Public and Dummy. When you enter the Safe area, a Safe partition designation replaces the Dummy partition designation. During this replacement process, the fixed DiskOnKey disconnects from and reconnects to the operating system.

Most of the deposit actions require the DiskOnKey to be reconnected to the system in order to notify the operating system of its new state. The SDK will perform the reconnection automatically and will block the events from reaching the SDK user. These methods can be executed either synchronously or asynchronously (recommended for Win32 non-console applications) depending on the *bBlock* value passed to them.

If *bBlock* is true, the method will run synchronously and return to the user after the DiskOnKey is reconnected. If *bBlock* is false, the method will run asynchronously and return to the user before the DiskOnKey is disconnected. A different thread is opened for the disconnect, which may result in an error being received from the callback.

The **IDOKOpHandler*** callback interface is passed to methods that can run synchronously or asynchronously. The progress and result of these methods will call back the appropriate method in the **IDOKOpHandler** callback derived object. The behavior of the callback will be the same whether using the methods, running synchronously or asynchronously. Passing this interface to the methods is not mandatory.

Table 5-7: Deposit API

Method	Description	Page
getHint	Returns the deposit password hint.	59
isInSafePartition	Verifies if any application is logged into the Safe partition.	60
supportDeposit	Verifies if this DiskOnKey supports the deposit API.	61
safePartitionExist	Verifies if a Safe partition exists on this DiskOnKey.	62
getPartitionsSize	Returns each partition size, in bytes.	63
configureDiskPartitions	Configures and formats the DiskOnKey using the **depositConfig** parameters.	64
changeDepositPassword	Changes the password required to login to the Safe partition.	66
enterSafePartition	Logs in to the Safe partition.	67
exitSafePartition	Logs out of the Safe partition.	69
minPartitionSize	Returns the minimum partition sizes for the Public, Safe and Dummy partitions.	70
CDOK Class Methods	CDOK Class Methods API.	28
Cookie API	CDOK Cookie API.	47
Cryptography API	CDOK Cryptography API.	71

5.7.1 getHint

Description

Returns the deposit password hint.

Syntax

```
DOK_STATUS getHint(char* szHint);
```

Parameters

szHint

[out] Returns the password hint.

Return Values

If the function succeeds, the return value is **DOK_OK** and *szHint* contains the password hint.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. The deposit API saves a 32Byte hint (for the deposit password) in the DiskOnKey device.
2. The **getHint** method enables the user to retrieve the hint from the DiskOnKey.
3. The hint is saved in the cookie area.
4. It is not protected like the deposit password.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , changeDepositPassword

5.7.2 isInSafePartition

Description

Verifies if any application is logged into the Safe partition.

Syntax

```
DOK_STATUS isInSafePartition(bool &bResult);
```

Parameters

`bResult`

[out] Specifies if the application is logged into the Safe partition.

Return Values

If the function succeeds, the return value is **`DOK_OK`**. *`bResult`* is true if the application is logged into the Safe partition. *`bResult`* is false if the application is not logged into the Safe partition.

If the function fails, the return value is != **`DOK_OK`**.

For detailed error information, refer to Error Codes.

Remarks

Since the method returns the DiskOnKey status, even if another application is logged in to the Safe partition, the method will return *`bResult`* = true.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **`IDeposit.h`**; included in **`DOK.h`**.

See Also:

CDOK Object, Deposit API , safePartitionExist

5.7.3 supportDeposit

Description

Verifies if this DiskOnKey supports the deposit API.

Syntax

```
DOK_STATUS supportDeposit(bool &bResult);
```

Parameters

bResult

[out] Specifies whether or not this device supports the deposit API.

Return Values

If the function succeeds, the return value is **DOK_OK** and *bResult* is true if this device supports the deposit methods. *bResult* is false if this device does not support the deposit methods.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

The deposit API is supported by DiskOnKey FW version 2.51 and higher.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , isInSafePartition

5.7.4 safePartitionExist

Description

Verifies if a Safe partition exists on this DiskOnKey.

Syntax

```
DOK_STATUS safePartitionExist(bool &bResult);
```

Parameters

bResult

[out] Specifies if a Safe partition exists on this device.

Return Values

If the function succeeds, that is the return value is **DOK_OK** and *bResult* is true if a Safe partition exists on this device. *bResult* is false if a Safe partition does not exist on this device.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

If there is no Safe partition on the DiskOnKey, it can be created using the configureDiskPartitions method.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , isInSafePartition , enterSafePartition , exitSafePartition

5.7.5 getPartitionsSize

Description

Returns each partition size in bytes.

Syntax

```
DOK_STATUS getPartitionsSize(unsigned long& ulPublicByte, unsigned
long& ulSafeByte, unsigned long& ulDummyByte);
```

Parameters

ulPublicByte

[out] Returns the `Public` partition size, in bytes.

ulSafeByte

[out] Returns the `Safe` partition size, in bytes.

ulDummyByte

[out] Returns the `Dummy` partition size, in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK**. *ulPublicByte* contains the Public partition size in bytes, *ulSafeByte* contains the Safe partition size in bytes and *ulDummyByte* contains the Dummy partition size in bytes.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

In the removable DiskOnKey, there is no Dummy partition, and therefore *ulDummyByte* contains 0.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , minPartitionSize

5.7.6 configureDiskPartitions

Description

Configures and formats the DiskOnKey using the **depositConfig** parameters.

Syntax

```
DOK_STATUS configureDiskPartitions(DepositConfig &depositConfig,
IDOKOpHandler* pCallback, bool bBlock);
```

Parameters

depositConfig

[in] Specifies the format configuration structure, detailed in **DOKSDK_DEF.h**. Refer to Data Types.

pCallback

[out] The callback receives the method results.

bBlock

[in] Specifies if the method will block for DiskOnKey reconnection. If *bBlock* is true, the method runs synchronously and blocks until the DiskOnKey reconnects to the operating system. If *bBlock* is false, the method runs asynchronously and without blocking. The result is passed by the callback API.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pCallback* also specifies whether the operation succeeded or failed.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The **configureDiskPartitions** method configures and formats the DiskOnKey partitions. The programmer can configure both the Deposit and Dummy partitions or delete one.
2. The parameters required for the partition configuration are all in the **DepositConfig** structure that is passed to the method.
3. Before performing the operation, the SDK locks the device partitions. If there is an open handle to one of the partitions, (an open file or application) the partition lock fails and the method returns a **MEDIA_LOCK_FAIL** error.
4. The partition lock is performed for safety reasons, before formatting the DiskOnKey.
5. Refer to the Deposit API for more information about running the method asynchronously.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in `IDeposit.h`; included in `DOK.h`.

See Also:

CDOK Object, Deposit API , supportDeposit

5.7.7 changeDepositPassword

Description

Changes the password required to login to the Safe partition.

Syntax

```
DOK_STATUS changeDepositPassword(const char* szOldPassword, const
char* szNewPassword, const char* szNewHint);
```

Parameters

szOldPassword

[in] Specifies the previous deposit password.

szNewPassword

[in] Specifies the new deposit password.

szNewHint

[in] Specifies the new deposit hint.

Return Values

If the function succeeds, the return value is **DOK_OK**. The password required to login to the Safe partition has been successfully changed, and the password hint has been updated.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

The maximum password size is 16Bytes and the maximum hint size is 32Bytes.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , getHint

5.7.8 enterSafePartition

Description

Logs in to the Safe partition.

Syntax

```
DOK_STATUS enterSafePartition(const char* szPassword, IDOKOpHandler*
pCallback, bool bBlock);
```

Parameters

szPassword

[in] Specifies the deposit password.

pCallback

[out] The callback receives the method results.

bBlock

[in] Specifies if the method will block for DiskOnKey reconnection. If *bBlock* is true, the method runs synchronously and blocks until the DiskOnKey reconnects to the operating system. If *bBlock* is false, the method will run asynchronously without blocking. The result is passed by the callback API.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pCallback* specifies whether the operation succeeded or failed.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method logs in to the DiskOnKey Safe area. The Safe area file system appears and the user is able to copy files from or to it.
2. In a fixed DiskOnKey, the Public and the Safe partitions will both be visible. The Safe partition replaces the Dummy partition.
3. In a removable DiskOnKey, the Safe partition replaces the Public partition.
4. The Safe area is closed, either by calling the **exitSafePartition** method or by unplugging the DiskOnKey from the computer.
5. Before performing the operation, the SDK locks the device partitions. If there is an open handle to one of the partitions (an open file or application), the partition lock fails and the method returns a **MEDIA_LOCK_FAIL** error.
6. Refer to Deposit API for more information about running the method asynchronously.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in `IDeposit.h`; included in `DOK.h`.

See Also:

CDOK Object, Deposit API , exitSafePartition , isInSafePartition

5.7.9 exitSafePartition

Description

Logs out of the Safe partition.

Syntax

```
DOK_STATUS exitSafePartition(IDOKOpHandler* pCallback, bool bBlock);
```

Parameters

pCallback

[out] The callback receives the method results.

bBlock

[in] Specifies if the method will block for DiskOnKey reconnection. If *bBlock* is true, the method will run synchronously and will block until the DiskOnKey will reconnect to the operating system. If *bBlock* is false, the method will run asynchronously and won't block. The result will be passed by the callback API.

Return Values

If the function succeeds, the return value is **DOK_OK**, *pCallback* also specifies whether the operation succeeded or failed.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method closes the DiskOnKey Safe partition.
2. In a fixed DiskOnKey, the Public and the Dummy partitions will both be visible. The Dummy partition replaces the Safe partition.
3. In a removable DiskOnKey, the Public partition replaces the Safe partition.
4. Before performing the operation, the SDK locks the device partitions. If there is an open handle to one of the partitions (an open file or application), the partition lock fails and the method returns a **MEDIA_LOCK_FAIL** error.
5. See the Deposit API for more information about running the method asynchronously.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , enterSafePartition , isInSafePartition

5.7.10 minPartitionSize

Description

Returns the minimum partition sizes for the Public, Safe and Dummy partitions.

Syntax

```
DOK_STATUS minPartitionSize(unsigned long& ulPartitionMinByte,
unsigned long& ulDummyMinByte);
```

Parameters

ulPartitionMinByte

[out] Returns the minimum Public partition size, in bytes.

ulDummyMinByte

[out] Returns the minimum Dummy partition size, in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK.** *ulPartitionMinByte* contains the minimum Public and Safe partition size, in bytes, and *ulDummyMinByte* contains the minimum Dummy partition size, in bytes.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. This method retrieves the minimum sizes of the DiskOnKey partitions.
2. The DiskOnKey cannot create a partition that is smaller than the minimum partition size. This is due to hardware limitations. An attempt to create a smaller partition will result in an error being returned.
3. The Public partition and the Safe partition have the same minimum size.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **IDeposit.h**; included in **DOK.h**.

See Also:

CDOK Object, Deposit API , getPartitionsSize

5.8 Cryptography API

The DiskOnKey enables PKI-based authentication and signing in a highly secure environment.

The SDK provides several APIs to enable the use of the following DiskOnKey cryptography capabilities:

- Supporting the DES/3 symmetric algorithm that enables encrypting and decrypting data buffers and files.
- Providing message and file digesting that enable signing files and data buffers.
- Performing PKI challenge response authentication.

The DESX encryption algorithm, utilized by the DiskOnKey encryption and decryption functions, is selected based on the customer profile (under some countries' limitations). The **ICrypto** interface enables an application to utilize the cryptographic capabilities of the DiskOnKey. It enables you to employ PKI, encrypt or decrypt data, and sign a file or buffer. All the cryptographic actions are executed on the DiskOnKey.

Every DiskOnKey has both a Private and a Public key. The Private key is unique to a specific DiskOnKey. The Public key is accessible to anyone who wishes to encrypt their data, however, only the DiskOnKey-unique Private key can decrypt the data.

Table 5-8: Cryptography API

Method	Description	Page
GetPublicKey	Returns the DiskOnKey Public key.	72
RandomChallenge	Decrypts a random challenge with the DiskOnKey Private key	73
RSAVerifyRandomChallenge	Verifies the decrypted random challenge by comparing it to the random challenge using an RSA algorithm.	74
MakeRandom	Generates a random number.	75
EncryptBuffer	Encrypts a data buffer using a DES/3 encryption algorithm.	76
DecryptBuffer	Decrypts a buffer using a DES/3 encryption algorithm.	77
GetEncryptionAlgorithm	Returns the encryption algorithm utilized by the DiskOnKey.	78
digitalSignature	Generates a digital signature for a text or binary file.	79
digitalSignature	Generates a digital signature for a buffer.	80
CDOK Class Methods	CDOK Class Methods API.	28
Cookie API	CDOK Cookie API.	47
Deposit API	CDOK Deposit API.	58

5.8.1 getPublicKey

Description

Returns the DiskOnKey Public key.

Syntax

```
DOK_STATUS getPublicKey(BYTE *pPublicKey);
```

Parameters

pPublicKey

[out] Holds the 64Byte DiskOnKey Public key.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pPublicKey* contains the 64Byte DiskOnKey Public key.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The DiskOnKey Public key is unique for each DiskOnKey. It is created during the DiskOnKey manufacturing procedure.
2. The DiskOnKey Public key size is 64Byte.
3. There is no way to change the DiskOnKey Public key.

Requirements

DiskOnKey FW Version: Requires 2.01 or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , RSAVerifyRandomChallenge, RandomChallenge

5.8.2 randomChallenge

Description

Decrypts a random challenge with the DiskOnKey Private key.

Syntax

```
DOK_STATUS randomChallenge(BYTE* pRandomChallenge, BYTE*
pDecryptedRandomChallenge );
```

Parameters

pRandomChallenge

[in] Specifies the 63Byte random challenge.

pDecryptedRandomChallenge

[out] Receives the 64Byte decrypted random challenge.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pDecryptedRandomChallenge* contains the decrypted random challenge.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The user creates a 63Bytes random number (**ICrypto::makeRandom** can be used) and passes it to the DiskOnKey. The DiskOnKey decrypts the random number using its Private key, and returns the 64Bytes decrypted random number to the user.

   Note: The buffers have different sizes:

   - Random challenge: 63Bytes (passed by the user to the DiskOnKey).
   - Decrypted random challenge: 64Bytes (returned by the DiskOnKey to the user).

2. The random challenge and the decrypted random challenge may be authenticated using the **ICrypto::RSAVerifyRandomChallenge** method.

Requirements

DiskOnKey FW Version: Requires 2.51 or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , RSAVerifyRandomChallenge

5.8.3 RSAVerifyRandomChallenge

Description

Verifies the decrypted random challenge by comparing it to the random challenge using an RSA algorithm.

Syntax

```
void RSAVerifyRandomChallenge(BYTE* pPublicKey, BYTE*
pRandomChallenge, BYTE* pDecryptedRandomChallenge);
```

Parameters

pPublicKey

[in] Specifies the Public key.

pRandomChallenge

[in] Specifies the 63Byte random challenge.

pDecryptedRandomChallenge

[in] Specifies the 64Byte decrypted random challenge.

Return Values

If the function succeeds, that is, the random challenge authenticated successfully, the return value is **DOK_OK**.

If the function fails, that is, the random challenge did not authenticate successfully, the return value is **DOK_ERR**.

For detailed error information, refer to Error Codes.

Remarks

1. This method will authenticate the decrypted random challenge (returned from the DiskOnKey) by encrypting the decrypted random challenge using the DiskOnKey Public key and then comparing it to the untouched random challenge.

   Note: The buffers have different sizes:

   - Random challenge: 63Bytes (passed by the user to the DiskOnKey).
   - Decrypted random challenge: 64Bytes (returned by the DiskOnKey to the user).

2. The **randomChallenge** function is used to decrypt the **random challenge.**
3. The **RSAVerifyRandomChallenge** function does not require the DiskOnKey to be connected.

Requirements

DiskOnKey FW Version: None.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , randomChallenge

5.8.4 makeRandom

Description

Generates a random number.

Syntax

```
void makeRandom(BYTE *pRand, int nRandByteSize);
```

Parameters

pRand

[out] Holds the random number generated by the SDK.

nRandByteSize

[in] Specifies the random number size, in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK** and a random number has been successfully generated.

If the function fails, the return value is != **DOK_OK** and an error is returned.

For detailed error information, refer to Error Codes.

Remarks

1. This method generates a pseudo random number, using the system time as a seed.
2. The random number may be used for a random challenge response.
3. The **makeRandom** function does not require the DiskOnKey to be connected, in order to work.

Requirements

DiskOnKey FW Version: None.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , randomChallenge

5.8.5 encryptBuffer

Description

Encrypts a data buffer using a DES/3 encryption algorithm.

Syntax

```
DOK_STATUS encryptBuffer(BYTE* pKey, BYTE* pBuffer, BYTE*
pEncrypBuffer, int nBufferByteSize);
```

Parameters

pKey

[in] Specifies the 24Byte key.

pBuffer

[in] Specifies the buffer containing the data to encrypt.

pEncrypBuffer

[out] Receives the encrypted buffer.

nBufferByteSize

[in] Specifies the size of the *pEncrypBuffer* buffer, in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pEncrypBuffer* contains the encrypted buffer.

If the function fails, the return value is != **DOK_OK** and an error is generated.

For detailed error information, refer to Error Codes.

Remarks

1. This function supports the Scramble, DES and DES/3 encryption algorithms.
2. Use the GetEncryptionAlgorithm function to check which specific algorithm, specified at the time of manufacture, is supported by DiskOnKey.
3. For DES/3, the buffer size for decryption should be specified as a multiple of 8. Otherwise, the buffer size should be specified as a multiple of 512.

Requirements

DiskOnKey FW Version: Requires 3.x or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , decryptBuffer

5.8.6 decryptBuffer

Description

Decrypts a buffer using a DES/3 encryption algorithm.

Syntax

```
DOK_STATUS decryptBuffer(BYTE* pKey, BYTE* pBuffer, BYTE*
pDecrypBuffer, int nBufferByteSize);
```

Parameters

pKey

[in] Specifies the 24Byte key.

pBuffer

[in] Specifies the buffer containing the data to decrypt.

pDecrypBuffer

[out] Receives the decrypted buffer

nBufferByteSize

[in] Specifies the size of the *pDecrypBuffer* buffer, in bytes.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pDecrypBuffer* contains the decrypted buffer.

If the function fails, the return value is != **DOK_OK** and an error is generated.

For detailed error information, refer to Error Codes.

Remarks

The algorithm used in these methods (encrypt/decryptBuffer) is symmetric, hence the same key should be used for both encryption and decryption.

Requirements

DiskOnKey FW Version: Requires 3.x or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , encryptBuffer

5.8.7 getEncryptionAlgorithm

Description

Returns the encryption algorithm utilized by the DiskOnKey.

Syntax

```
DOK_STATUS getEncryptionAlgorithm (eEncryptAlgorithm& alg);
```

Parameters

alg

[out] Returns the encryption algorithm utilized by the DiskOnKey.

Return Values

If the function succeeds, the return value is **DOK_OK**, *alg* contains an enumeration of the encryption algorithm utilized by the DiskOnKey.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The DiskOnKey can use one of the following encryption algorithms:
   - **DES**
   - **DES3**
   - **Scramble**
2. Use the GetEncryptionAlgorithm function to check which specific algorithm, specified at the time of manufacture, is supported by DiskOnKey.
3. DES/3 is not implemented, by default, because some governments consider devices with DES/3 abilities to be an encryption machine, and impose specific limitations on those devices. To avoid dealing with legal issues concerning DES/3, the encryption algorithm is defined for the DiskOnKey during the DiskOnKey manufacturing process, taking into account the DiskOnKey customer’s country.

Requirements

DiskOnKey FW Version: Requires 3.x or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , getPublicKey

5.8.8 digitalSignature

Description

Generates a digital signature for a text or binary file.

Syntax

```
DOK_STATUS digitalSignature(const char* szFilePath, BYTE*
pCertificate, BYTE* pSignature);
```

Parameters

szFilePath

[in] Specifies the absolute path to the file.

pCertificate

[in] Specifies the 43Byte certificate.

pSignature

[out] Receives the 64Byte digital signature for a file.

Return Values

If the function succeeds, the return value is **DOK_OK** and *pSignature* contains the digital signature for a file.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The **digitalSignature** method generates a digital signature (digest) of the input file.
2. The 43Byte certificate, passed to the method, is used to build a unique signature.
3. A digital signature is used to authenticate files for changes. If a file is signed, a change in the file, even in one byte, will generate a different signature. (Both signatures are created using the same certificate.)

Requirements

DiskOnKey FW Version: Requires 3.x or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , digitalSignature

5.8.9 digitalSignature

Description

Generates a digital signature for a buffer.

Syntax

```
DOK_STATUS digitalSignature(BYTE* pBuffer, int nBufferByteSize, BYTE*
pCertificate, BYTE* pSignature);
```

Parameters

pBuffer

[in] Specifies the buffer.

nBufferByteSize

[in] Specifies the buffer size, in bytes.

pCertificate

[in] Specifies the 43Byte certificate.

pSignature

[out] Receives the 64Byte digital signature for a buffer.

Return Values

If the function succeeds, the return value is **DOK_OK**, and *pSignature* contains the digital signature for a buffer.

If the function fails, the return value is != **DOK_OK**.

For detailed error information, refer to Error Codes.

Remarks

1. The **digitalSignature** method generates a digital signature (digest) for a data buffer.
2. The 43Byte certificate, passed to the method, is used to build a unique signature.

Requirements

DiskOnKey FW Version: Requires 3.x or later.

Header: Declared in **ICrypto.h**; included in **DOK.h**.

See Also:

CDOK Object, Cryptography API , digitalSignature

5.9 Utilities

CSDKUtils contains several static methods, useful in development.

Table 5-9: Utilities

Method	Description
maskToDriveLetters	Receives the drive letters mask and returns the drive letters string.
driveLetterToMask	Receives the drive letters string and returns the drive letters mask.
osVersion	Returns the operating system version.
trimRight	Removes white space to the right of a given string.
trimLeft	Removes white space to the left of a given string.
trimRightLeft	Removes white space to the right and left of a given string.
errorString	Receives an error message enumerator and returns an error message string. Refer to Error Codes.

6 Troubleshooting

The following sections provide a listing of the error codes that may be encountered and their meaning, and the URL for customer support information.

6.1 Introduction

For information regarding troubleshooting, refer to **http://www.diskonkey.com/**.

6.2 Error Codes

The error codes are defined as an enumerator in **DEVICEPARAMS_H**.

Table 6-1: Error Codes

Type	Name	Description
General Errors	DOK_OK	Status OK.
	DOK_ERR	General error.
	ACCESS_ERR	SDK cannot access a disk device.
	COMM_FAIL	Connection to device failed.
	DOK_INTERNAL_ERR	Device internal error.
	VERIFICATION_ERR	Verification error.
	LUN_SWAP_FAIL	Error in swap between LUNs.
	MEDIA_LOCK_FAIL	If there is an open handle to one of the partitions (an open file or application), the partition lock fails and the method returns a **MEDIA_LOCK_FAIL** error.
States	DOK_NOT_CONNECTED	The device is not connected.
	AUTHENTICATION_ERR	SDK cannot authenticate a disk device.
	DOK_CHANGED	Change in device parameters.
	DOK_NOT_READY	Device status is not ready.
	DOK_UCL	The device is the UCL version.
	NO_SAFE_PARTITION	No safe partition.
Cmd	FORMAT_FAIL	The format function failed.
	CMD_NOT_SUPPORTED	The DiskOnKey firmware version does not support the attempted function.
Internal	DOK_DOESNOT_DETACH	No need to attach the device.
	DEVICE_EXISTS	The device already exists in the SDK.
	DOK_TOO_SMALL	The device is too small to format.
	DOK_TOO_BIG	The device is too big to format.
Hidden Area	PAGE_FULL	Hidden page is full.

	DOK_FULL	There is no space left in the hidden area.
	COOKIE_EXISTS	Hidden area data already exists.
	DATA_NOT_FOUND	The entry or section does not exist.
Exceptions	INIT_METHOD_CALLED_AGAIN	Cannot call **`init`** method more then once.
	NO_INIT_METHOD_CALLED	Must call **`init`** method to initialize the DiskOnKey SDK.
Method Parameters	BUFFER_TOO_SMALL	The buffer pointed to is not large enough.
	PARAMS_ERR	A null section or null entry has been entered.
	DRIVE_LETTER_ERR	The partition is not a DiskOnKey partition, or the SDK cannot recognize it.

6.3 Troubleshooting FAQ

For troubleshooting FAQ, refer to **http://www.diskonkey.com/**.

Appendix A: Data Types

The following structs are passed as parameters in the API methods.

A.1 DepositConfig

The following struct is defined in **DOKSDK_DEF_H**. Refer to configureDiskPartitions.

```
struct DepositConfig
{
     char szPaassword[DEPOSIT_PASSWORD_SIZE+1];
     char szHint[DEPOSIT_HINT_SIZE+1];
     union
     {
          FixDokConfig         fp;
          RemovableDokConfig   rp;
     };
};
```

A.2 FormatConfig

The following struct is defined in **DOKSDK_DEF_H**. Refer to format.

```
//for use in ICommon::format method
struct FormatConfig
{
     bool bReadMBR;
     bool bReadPartitionBootSector;
     bool bLBA;
     BYTE*     pMbr;
     BYTE*     pPartitionBootSector;
     bool bConfigCHS;

     struct partition
     {
          char szLabel[11];
          bool bBootable;
          int       nPartitionSectorSize;//if 0 - will take params
from mbr
          partition* pNextPartition;//should be null for last
partition
     }partitionList;
```

Bates Beg :	WDC0011514
Bates End :	WDC0011597
Confidential Designation :	Confidential - Attorneys' Eyes Only
File Type :	Adobe Portable Document Format
Date and Time Last Modified :	10/10/2002 12:45:00 PM
Date and Time Created :	10/10/2002 12:42:00 PM
Email From :	
Email To :	
Email CC :	
Email BCC :	

EXHIBIT 15

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INGENICO INC.,
Petitioner,

v.

IOENGINE, LLC,
Patent Owner.

IPR2019-00879

Patent 9,059,969

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

Regulations

TABLE OF EXHIBITS

Exhibit No.	Description
2003	Declaration of Kevin Butler, Ph.D.
2004	Dennis J. M. J. de Baar et al., *Coupling application design and user interface design*, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92), Penny Bauersfeld, John Bennett, and Gene Lynch (Eds.). ACM, New York, NY, USA, 259-266. DOI: https://doi.org/10.1145/142750.142806
2005-2017	Exhibit Numbers Not Used
2018	Ronald Rivest, Chaffing and Winnowing: Confidentiality without Encryption, (http://people.csail.mit.edu/rivest/chaffing-980701.txt)
2019	Complaint, *IOENGINE, LLC v. GlassBridge Enterprises, Inc. (formerly Imation Corp.*), No. 1:14-cv-01572-GMS, D.I. 1 (D. Del. Dec. 31, 2014)
2020	Complaint, *IOENGINE, LLC v. Interactive Media Corp*., No. 1:14-cv-01571-GMS, D.I. 1 (D. Del. Dec. 31, 2014)
2021	Verdict Form, *IOENGINE, LLC v. GlassBridge Enterprises, Inc. (formerly Imation Corp.*), No. 1:14-cv-01572-GMS, D.I. 202 (D. Del. Feb. 17, 2017)
2022	Verdict Form, *IOENGINE, LLC v. Interactive Media Corp*., No. 1:14-cv-01571-GMS, D.I. 160 (D. Del. Jan. 17, 2017)
2023-2028	Exhibit Numbers Not Used
2029	Complaint, *IOENGINE LLC v. PayPal Holdings, Inc.*, No. 1:18-cv-452-WCB, D.I. 1 (D. Del. Mar. 23, 2018)
2030	Answer and Counterclaim, *Ingenico Inc. v. IOENGINE LLC*, No. 1:18-cv-826-WCB, D.I. 12 (D. Del. Aug. 17, 2018)
2031	Scheduling Order, *Ingenico*, No. 18-826, D.I. 57, *PayPal*, No. 18-452, D.I. 49 (Jan. 28, 2019)
2032	Motion to Dismiss and Supporting Brief, *Ingenico*, No. 18-826, D.I. 26-27 (Oct. 9, 2018)
2033	Order setting hearing on Motion to Dismiss, *Ingenico*, No. 18-826, D.I. 43 (Nov. 29, 2018)
2034	Exhibit Number Not Used
2035	Ingenico Inc.'s and Ingenico Corp.'s Answer to Counterclaim, *Ingenico*, No. 18-826, D.I. 68 (Mar. 11, 2019)

Exhibit No.	Description
2036	Excerpt of Ingenico, Inc.'s April 12, 2019 First Amended Initial Contentions Cover Document in *Ingenico*, No. 18-826
2037-2039	Exhibit Numbers Not Used
2040	Excerpts of PayPal Holdings Inc.'s April 5, 2019 Initial Invalidity Contentions Cover Document in *PayPal*, No. 18-452
2041-2044	Exhibit Numbers Not Used
2045	Letter from U.S. Senators Thom Tillis and Christopher A. Coons to the Director of the U.S. Patent and Trademark Office, Apr. 9, 2019
2046	Letter to Judge Bryson regarding withdrawal of Ingenico's motion to dismiss, Ingenico, No. 18-826, D.I. 69 (Mar. 26, 2019)
2047-2061	Exhibit Numbers Not Used
2062	Curriculum vitae of Kevin Raymond Boyce Butler
2063-2068	Exhibit Numbers Not Used
2069	2018-03-23 Letter from Noah Leibowitz to Pentland Walcott
2070	Joint Claim Construction Chart, *IOENGINE LLC v. PayPal Holdings, Inc.*, No. 1:18-cv-452-WCB, D.I. 86 (D. Del. June 25, 2019)
2071	Exhibit Number Not Used
2072	Excerpts of Petition for *Inter Partes* Review, Ingenico Inc. v. IOENGINE, LLC, IPR2019-00416
2073	Excerpts of Petition for *Inter Partes* Review, Ingenico Inc. v. IOENGINE, LLC IPR2019-00584
2074	Highlighted copy of Petitioners Exhibit 1004, FujiFilm Software Quick Start Guide, Petition for Inter Partes Review, Ingenico Inc. v. IOENGINE, LLC IPR2019-0879
2075	Highlighted copy of Petitioners Exhibit 1012, FujiFilm Software Quick Start Guide, Petition for Inter Partes Review, Ingenico Inc. v. IOENGINE, LLC IPR2019-00879
2076	Excerpt of Decision Instituting *Inter Partes* Review, Paper 20, Ingenico Inc. v. IOENGINE, LLC, IPR2019-00416

I. INTRODUCTION

IOENGINE LLC ("IOENGINE" or "Patent Owner") respectfully submits this Preliminary Response together with the Expert Declaration of Kevin Butler, Ph.D. (Ex. 2003) under 35 U.S.C. § 313 and 37 C.F.R. § 42.107, responding to the Petition for *Inter Partes* Review (the "Petition") filed by Ingenico Inc. ("Ingenico" or "Petitioner") challenging claims 1-8, 10-16, 19-21, and 24-29 (the "Challenged Claims") of U.S. Patent No. 9,059,969 (the "'969 Patent"). IOENGINE is the assignee of all rights and interest in the '969 Patent.

The Petition should be denied and *inter partes* review should not be instituted for at least the following reasons: ***First***, the Petition alleges four grounds for unpatentability but does not demonstrate a reasonable likelihood of prevailing on any. As discussed in detail below, none of the cited references disclose multiple claim elements that appear in the independent Challenged Claims. Further, the dependent Challenged Claims have additional limitations which also are not disclosed by the cited references.

In addition, as discussed below with respect to Grounds 2-4, the obviousness arguments in the Petition fail because they do not provide adequate rationale or motivation for combining the cited references to arrive at the Challenged Claims.

Second, the Board should exercise its discretion to deny the Petition under 35 U.S.C. 314(a). The Petition stems from an infringement action that IOENGINE filed against PayPal Holdings, Inc. ("PayPal"). Ex. 2029. After it received an indemnification request from PayPal, Ingenico injected itself into the litigation by filing a declaratory judgment action. Ex. 1019 ¶¶ 5, 7-9. Ingenico and its indemnitee then proceeded to file a set of serial IPRs. Altogether, Petitioner and its indemnitee have filed a staged sequence of ***twelve*** IPR petitions against the three patents-at-issue in the district court litigation, with ten of the twelve petitions, including this Petition, not filed until nearly the eve of Petitioner's indemnitee's one-year statutory bar date. The resulting unnecessary burden and inefficiency is highlighted by the fact that the district court proceedings have been consolidated before the Honorable William C. Bryson, of the United States Court of Appeals for the Federal Circuit, sitting by designation, with both cases sharing all dates until trial. Ex. 2031. In addition, both Ingenico and PayPal rely for their invalidity cases in the district court on the same references (among others) that are relied on in the Petition. And, because Ingenico waited until more than a year after the litigation against its indemnitee PayPal was filed, and just two weeks from the one-year statutory bar date based on service of that complaint, the final decision in this IPR, if instituted, would not be expected until trials in both district court litigations have concluded.

The problem of serial, vexatious, and inefficient IPR proceedings has become acute. In April 2019, Senators Thom Tillis and Christopher A. Coons of the Senate Judiciary Committee wrote to Director Iancu recommending that the Board take action to curb such abusive tactics. Ex. 2045. This case is a poster-child for such abuse. Instead of the single, coordinated, efficient district court proceeding pending before Circuit Judge Bryson—into which Petitioner chose to insert itself, and in which the same invalidity theories are raised—Petitioner and its indemnitee seek to multiply the proceedings twelve-fold. The Board should exercise its discretion to preserve efficiency and deny institution under these circumstances.

II. RELATED PROCEEDINGS

The current set of proceedings originated more than a year ago, on March 23, 2018 with IOENGINE's suit against PayPal (the "PayPal Action"). Ex. 2029. After receiving an indemnification request from PayPal, Ingenico filed a declaratory judgment action against IOENGINE on June 1, 2018 (the "Ingenico Action"). Ex. 1019 ¶¶ 5, 7-9. Ingenico acknowledged in its Complaint that the PayPal Action "triggered an indemnity request by PayPal to Ingenico," and relied on this allegation to establish standing. Ex. 1019 ¶ 9. The district court proceedings are now consolidated before the Honorable William C. Bryson, coordinated for fact and expert discovery and claim construction, and sharing all dates until separate trials in July and August, 2020. Ex. 2031. Both cases involve the same three related asserted IOENGINE patents: U.S. Patent Nos. 8,539,047; 9,059,969; and 9,774,703.

The'047 Patent has previously been tried twice to jury verdict for Patent Owner. Exs. 2019, 2021, 2020, 2022. Two different juries found claims of the '047 Patent infringed and not invalid, and awarded damages to IOENGINE. *Id.*

In addition to this Petition challenging the '969 Patent, Ingenico's indemnitee has filed two additional petitions challenging the '969 Patent. Altogether, Ingenico and PayPal have filed twelve petitions challenging the patents-at-issue in the Ingenico and PayPal Actions, as listed in the table below.

Case	**Filer**	**Filed**	**Patent**
IPR2019-00416	Ingenico	12/17/2018	'047
IPR2019-00584	Ingenico	1/22/2019	'703
IPR2019-00879	***Ingenico***	***3/25/2019***	***'969***
IPR2019-00884	PayPal	3/29/2019	'047
IPR2019-00885	PayPal	3/29/2019	'047
IPR2019-00886	PayPal	3/29/2019	'047
IPR2019-00887	PayPal	3/29/2019	'047
IPR2019-00906	***PayPal***	***4/4/2019***	***'969***
IPR2019-00907	***PayPal***	***4/4/2019***	***'969***
IPR2019-00929	Ingenico	4/5/2019	'703
IPR2019-00930	PayPal	4/8/2019	'703
IPR2019-00931	PayPal	4/8/2019	'703

III. THE '969 PATENT

The '969 Patent describes a tunneling client access point ("TCAP") that communicates with both an access terminal (*e.g.*, a cellular telephone or computer) and a remote network device (*e.g.*, a server). Ex. 1001 abstract, 1:19-25, 2:39-51, 3:41-4:30, 18:12-14, Figs. 1, 9-10.

To prevent the terminal from accessing information sent between the TCAP and the server, the TCAP secures the data such that "if data moving out of the TCAP and across the [access terminal] were captured at the [access terminal], such data would not be readable." *Id*. at 13:1-4, 27:28-28:15, Fig. 10. For example, the TCAP preferably includes a Cryptographic Server Module, which can be used to "encrypt all data sent through the access terminal based on the TCAP's unique ID and user's authorization information." *Id.* at 28:12-15. This is not exemplary or permissive language; it is a concept that is captured in the very title of the '969 Patent. *See* '969 Patent at Title ("Apparatus, method and system for a ***tunneling*** client access point "), Abstract ("The disclosure details the implementation of a ***tunneling*** client access point (TCAP) ***that is a highly secure***, portable, power efficient storage and data processing device. The TCAP ***'tunnels' data through*** an access terminal's (AT) input/output facilities."), 2:39-51, 3:41-4:30; Ex. 2003 ¶ 24.

Users interact with the TCAP through an interactive user interface ("IUI") presented by the terminal. Ex. 1001 abstract, 2:39-46, 3:57-62, 17:51-18:3. The IUI "provides a facility through which users may affect, interact, and/or operate a computer system," and is presented to the user on the terminal's output component. Ex. 1001 at 26:19-20, 6:64-67, 7:12-13, 7:45-47, 8:37-41, 9:5-10:46, 12:33-62, 26:7-14, Figs. 4, 5-8, 10; Ex. 2003 ¶ 25.

Applications of the TCAP include, for example, improving the security of accessing remote data, encrypted communication, and secure purchasing, payment and billing. Ex. 1001 at abstract, 2:49-51, 3:64-4:30, 5:32-54, 7:9-8:45, 8:63-9:1, 10:44-61, 12:55-13:27, 19:38-20:24, 27:28-28:15, Figs. 2, 4, 9-10.

A. <u>Pre-AIA Statutes Apply</u>

Petitioner's argument that post-AIA statutes apply seeks an advisory opinion because the prior art status of Petitioner's references does not depend on which statute applies. Moreover, the '969 Patent was filed as a pre-AIA application, and the file history, including office actions and the Notice of Allowability note its pre-AIA status. Ex. 1018 at 115, 161, 167, 210. Finally, the specification fully supports claim 9, which is the only claim Petitioner alleges lacks written description. *See* Ex. 1001, 19:12-15 ("A user programs table 919c includes fields such as, but not limited to: system programs, organization programs, ***programs to be synchronized***, and/or the like."), 27:3−9 (same), 28-31 ("If synchronization is specified 470, then the TCAP will ***provide and receive*** updated data ***to and from*** the backend servers…").

IV. POSITA

A person of ordinary skill in the relevant art ("POSITA") would be a person with a Bachelor of Science degree in Computer Science, or related discipline, and two to three years of experience in developing, implementing, or deploying systems for the encryption of data on a portable device. Ex. 2003 ¶¶ 11-13.

V. ASSERTED GROUNDS AND REFERENCES

Petitioner asserts four grounds for invalidity as summarized on page 5 of the Petition. The art relied on in the Petition is as follows:

A. Iida

Iida describes a digital camera designed to be a lower-cost, rental alternative to more expensive digital cameras that included on-board LCDs. Ex. 1003 [0005]-[0007], [0010]-[0012], [0030], [0036]. The camera can be used to place orders for photographic prints from a remote photo lab. *Id.* abstract, [0004], [0033]-[0036], [0072], [0076], [0138].

Iida's rental camera communicates with a portable communication apparatus or image display apparatus possessed by the user. *Id.* [0014], [0032], [0040], [0041], [0074]. The portable apparatus used with Iida's digital camera can display a menu image, but the menu image is not interactive. *Id.* [0068], [0083]. The user can press number keys on the portable apparatus, whereupon the number pressed on the keypad is transmitted to the camera. *Id.* [0068], [0084], [0093], [0098], Fig. 4C. The camera determines what action to take or whether to generate a new static image for display by the portable apparatus. *Id.* [0084], [0085], [0098]-[0107], Figs. 4A, 4C. Unlike the terminal described in the '969 Patent, the portable apparatus in Iida does not take any action responsive to the user pressing a number key—it simply transmits the inputted number to the camera. *Id.* [0068], [0084]-[0085], [0097]-[0099], [0130]-[0132]; Ex. 2003 ¶¶ 28-30.

Iida also refers to a personal computer ("PC") which the user requires in order to store and/or print images from the camera. Ex. 1003, [0008], [0067], [0106]-[0107]. There is no indication in Iida that the PC functions as the portable apparatus or is involved in any communication beyond the user's premises.

B. The Fuji Guide

The FUJIFILM Software Quick Start Guide ("Fuji Guide") describes how to install and use certain software for viewing images on a PC. Ex. 1004 *passim*. The Fuji Guide is undated and contains no indication of when, if ever, it was published. The only evidence that Petitioner provides to show publication of the Guide is in the Declaration of Paul Widener, who claims that, "to the best of [his] knowledge" (*i.e.*, from memory of an event eighteen years ago), he purchased a Fuji FinePix 6800 Zoom camera in June of 2001 and the camera was accompanied by the Fuji Guide. Ex. 1005 ¶¶ 4, 14.

Petitioner's reliance on Mr. Widener's alleged receipt of the Fuji Guide with a purchase of a Fuji camera fails to meet the standard of proving that the Fuji Guide constitutes a printed publication before the priority date of the '969 Patent. The proper inquiry is whether the reference was "sufficiently accessible to the public interested in the art" before the relevant date. *In re Cronyn*, 890 F.2d 1158, 1160

(Fed. Cir. 1989). This requires that "persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it." *Acceleration Bay, LLC v. Activision Blizzard Inc.*, 908 F.3d 765, 772, 774 (Fed. Cir. 2018) (citations omitted). Merely showing that a product was on sale is not sufficient to prove that an accompanying product manual was a printed publication before the relevant date. *Lantz Med., Inc. v. Bonutti Research, Inc.*, IPR2015-00995, Paper 11 (PTAB Oct. 21, 2015), at 8-9.

In this case, Petitioner has presented no evidence that a POSITA would have been able to find the Fuji Guide at any time relevant to this case. There is no evidence, for example, that the Fuji Guide appeared in any index or catalog or was locatable on the Internet during the relevant time period. *See, e.g., Acceleration Bay*, 908 F.3d at 773-74. Even accepting Mr. Widener's testimony, Petitioner provides no evidence that a POSITA would have known that the Fuji Guide would accompany the purchase of a camera, much less been able to seek it out.

There is no evidence that Mr. Widener sought out or requested the Guide; more likely he bought the camera not knowing what documentation would come with it. Mr. Widener does not say from whom purchased the camera or obtained the Fuji Guide. Ex. 1005 ¶¶ 2-16. His declaration is not enough to meet Petitioner's burden of proving that a POSITA, exercising reasonable diligence, could have searched for and found the Fuji Guide before March 23, 2004. *Acceleration Bay*, 908 F.3d at 772, 774.

Petitioner and its expert Mr. Geier purport to corroborate Mr. Widener's declaration by referring to metadata from a different manual that Petitioner allegedly found online. Petition 10 (referring to Ex. 1012). But the online manual is clearly not the same document as the one provided by Mr. Widener, as shown by the numerous discrepancies highlighted in the copies provided herewith. *See* Ex. 2074 (highlighted version of Ex. 1004), Ex. 2075 (highlighted version of Ex. 1012). Indeed, Petitioner's recently found, online manual appears to be an unfinished draft, since it goes up to page 19 before restarting at page 2, and is missing numerous passages that appear in Mr. Widener's version. *Id*. Furthermore, Petitioner provides no evidence of when Ex. 1012 was published. The metadata on which Mr. Geier relies pertains only to the "create date" and "modify date" of the document, which have no bearing on the publication date. Ex. 1002 ¶ 41. Indeed, date information within a document itself is inadmissible hearsay and insufficient to establish public accessibility. *Laird Techs. Inc. v. A.K. Stamping Co. Inc.*, IPR2017-02038, Paper 6 (PTAB Mar. 14, 2018), at 9-10. Even Mr. Geier admits that he is "unable to determine when Fuji web page with the [online version of the] guide was made publicly available." Ex. 1002 ¶ 41. Thus, Mr. Widener's testimony regarding when he obtained the Fuji Guide is uncorroborated.

Even if the Fuji Guide were proven to be prior art, it would not have been obvious to combine it with Iida. Although Iida refers to "a digital still camera 'FinePix 68007' manufactured by Fuji Photo Film Co., Ltd.," it makes no mention of the Fuji Guide. Ex. 1003 [0153]. Moreover, the software described in the Fuji Guide is installed on the user's personal computer ("PC"), which can be located in the user's home or home office. *See, e.g., id*. 2, 3, 5, 7, 33 (referring to "in-house LAN" or "home-office LAN"). The Fuji Guide contains no mention of connecting the camera to a portable apparatus or using the PC in a portable manner. The software described in the Guide does not allow the camera to communicate through the PC to a network or with a server. Instead, the software allows the user to save photos from the camera to a folder on the PC for later access, view thumbnails and full-size images on the screen of the PC, and perform image processing. Ex. 1004, 7 ("images stored on your PC"), 17-18.

C. <u>Shaffer</u>

Shaffer describes a system for controlling access to images created by scanning a customer's exposed film at a photofinishing location, and providing services related to those images. Ex. 1006, Title, Abstract; 1:26-31 ("[A] photography customer would drop off her exposed film at a photofinisher where it would be processed, photographically printed and optionally scanned at a high resolution. . . ."); 2:25-30 (method "includes the steps of: scanning a customer film image to generating high and low resolution digital versions of the image . . . ."). After the customer's film is scanned, the resulting digital images are processed to produce high and low resolution versions. *Id*. 3:12-17. The customer can store the high-resolution images at an image fulfillment center and the low-resolution versions locally on a floppy diskette. *Id*. 3:28-31, 3:42-48, 4:16-22, 5:4-6, 17-19, Fig. 1. The fulfillment center may provide prints and enlargements of the image or send the difference between the low and high resolution versions of the image so that the high-resolution image can be reconstructed. *Id*. 5:26-30, 5:44-50, 6:6-18; Ex. 2003 ¶ 35.

D. <u>Ford</u>

Ford is a 160-page collection of chapters and a partial chapter from a textbook on secure electronic commerce. The excerpts provided by Petitioner generally discuss cryptography, Internet security, and certificates. There is no identifiable connection between Ford and Iida, Shaffer or the Fuji Guide.

VI. CLAIM CONSTRUCTION

The parties agree that the claims should be accorded their ordinary and customary meaning as understood by one of ordinary skill in the art, in light of the intrinsic record pertaining to the patent. 37 C.F.R. § 42.100(b).

The meaning of a claim term is not evaluated in a vacuum, however, but "in the context of the entire patent, including the specification" and "the prosecution history." *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (citation omitted). In particular, "the specification 'is always highly relevant to the claim construction analysis [and] [u]sually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id.* at 1315 (citation omitted). Furthermore, "[t]he fact that [a particular characteristic] is 'repeatedly and consistently' used to characterize the invention strongly suggests that it should be read as part of the claim." *VirnetX, Inc. v. Cisco Sys., Inc.*, 767 F.3d 1308, 1318 (Fed. Cir. 2014).

A. IUI

The term "interactive user interface" ("IUI") appears in all independent Challenged Claims, and thus all Challenged Claims. The Board should construe it to mean "a presentation containing interface elements with which a user may interact to result in the terminal taking action responsively by modifying what is presented." Ex. 2003 ¶¶ 37-45.

The claims uniformly require the IUI to be presented on the access terminal, *id*. Abstract, 4:39-64, claims 1, 28, 29 (display on/by "the terminal output component"), Figs. 5-8. Furthermore, the IUI is "interactive" in the sense that the user may manipulate the IUI such that the device on which the IUI resides—the terminal—acts *responsively* to the user's input by modifying what is presented. *Id.* 9:37-48 (when user clicks button, interface "further unfurl[s]" to present options to access facilities and services, including a login button which takes user to a login screen), 10:16-23 (engaging the interface by dragging and dropping files), 10:64-11:4 (drag and drop), 11:13-31, 61-64 (unfurling interface by graphically opening can of soda), Figs. 5-8. In contrast, if the user's input were simply passed along to a different device, and the device presenting the interface behaved no differently regardless of the user's input, then the user would not be *interacting with* the interactive user interface. Ex. 2003 ¶ 40.

The specification explains that user interaction takes place by way of "computer interaction interface elements such as check boxes, cursors, menus, scrollers, and windows…." '047 Patent at 1:52-62; *see also id*. at Figs. 5-8 (showing various interaction interface elements of an IUI). The interaction interface elements are not simply examples or suggestions. It is these interaction interface elements that "allow for the display, execution, interaction, manipulation, and/or operation of program modules and/or system facilities through textual and/or graphical facilities," and "provide[] a facility through which users may affect, interact, and/or operate a computer system." *Id*. at 17:58-63; *see also* 10:33-47 (engaging an interface element to manipulate data), 11:14-18 (accessing help facilities by "engaging a help facility user interface element").

Indeed, a central purpose of the invention of the Patents-in-Suit is to allow users to interact with the TCAP by employing "traditional large user interfaces" that users "are already comfortable with," as opposed to then-existing portable computing devices, which had "uncomfortably small user interfaces." *Id.* Summary at 2:25-37. This makes the disclosed TCAP easy to use, as "at most it requires the user to simply plug the device into any existing and available desktop or laptop computer, through which, the TCAP can make use of a traditional user interface and input/output (I/O) peripherals…" *Id*. Summary at 2:37-46.

One problem with Petitioner's construction is that Petitioner's reference to "the computer" is ambiguous. Each Challenged Claim refers to multiple devices that could be considered computers. Ex. 1001 claims 1, 28, 29 (reciting a portable device, a terminal, and a communications network node). As the Petition ***admits***, however, in the '969 Patent "[a] number of the displays respond to text inputs from an input component to elicit ***a responsive action from the terminal***." Petition 13-14 (emphasis added). Thus, both parties agree that it is the terminal, not the portable device, that responds to user interaction. Likewise, the examples of the "baseline" IUIs in the '969 specification (Macintosh Aqua, Windows XP, X-Windows, etc.) all provide for the device presenting the user interface—*i.e.*, the terminal—to take responsive action. Although the claims require that the IUI may ultimately be used to cause the TCAP to execute code, Ex. 1001 at 6:64-67, 7:12-14, 7:45-47, 8:37-41, 9:5-10:46, 12:33-62, the claims require that it is presented on the terminal and the specification consistently and invariably describes the terminal as acting *responsively* to the user's input by modifying what is presented. Accordingly, the proper construction of interactive user interface should make clear that it is the "***terminal***"—not a generic "computer"—that takes action responsively as a result of user interaction to modify what is presented. Ex. 2003 ¶ 43.

Petitioner's proposed construction of "graphic user interface," which is a specific type of IUI, implicitly acknowledges the need for an IUI to have elements with which a user may interact. According to Petitioner, a graphic user interface is "a display with which ***a user may interact, facilitated by a graphic element displayed thereon***, to result in the computer taking action responsively." Petition 14-15 (emphasis added)[1]

Additionally, Petitioner's definition fails to specify what it means for the user to "interact" with the display and for the device presenting the interface to "take action responsively." The specification shows that what the inventor intended by an IUI was for the terminal to present one or more interface elements for the user to

[1] Petitioner attempts to limit IUIs to visual "displays." But the '969 Patent indicates that the IUI can comprise audio and tactile elements. Ex. 1001, 14:58-15:16. Patent Owner's definition uses the term "presentation" to account for this. In the district court proceeding, Petitioner has adopted Patent Owner's position and proposed "presentation" instead of "display" for its construction. Ex. 2070.

"engage" with, and then act responsively to modify the presentation of the interface. These are the "interaction interface elements" (*e.g.*, "check boxes, cursors, menus, scrollers, and windows") to which the specification refers. Ex. 1001 at 1:52-56, 8:19-21, 9:32-48, 10:19-23, 10:32-43, 10:62-11:4, 11:13-31, 11:61-64, Figs. 5-8; Ex. 2003 ¶ 44.

Accordingly, "interactive user interface" should be construed to mean "a presentation containing interface elements with which a user may interact to result in the terminal taking action responsively by modifying what is presented."

B. GUI

The term "graphic user interface" ("GUI") appears in Challenged Claim 21 of the '969 Patent and further limits the IUI of independent claim 1. Petitioner's proposed construction is "a display with which a user may interact, facilitated by a graphic element displayed thereon, to result in the computer taking action responsively." Petition 14-15. Petitioner claims that this is the "ordinary meaning" of the term, but it artificially broadens the term to try to include any interface that displays a "graphic element," ***even if the element is passive and the user does not interact with it***. Petition 14-15, 40-41 (applying the term to passive thumbnail images); Ex. 1003, [0095]-[0096], [0128]. Thus, Petitioner seeks to read out from claim 21 the "interactive user interface" of parent claim 1.

Since the GUI of claim 21 is a type of IUI as recited claim 1, a proper definition of GUI must incorporate the above definition of IUI. Accordingly, the definition of GUI should simply add the word "graphical" to the definition of IUI: "a presentation containing ***graphical*** interface elements with which a user may interact to result in the terminal taking action responsively by modifying what is presented." Ex. 2003 ¶ 46.

C. <u>"communications . . . to a communications network node through the terminal [network] communication interface" and "communication[s] . . . through . . . the terminal network interface to a communications network node"</u>

These terms appear in all independent Challenged Claims and thus all Challenged Claims. They should be construed as "communications sent through the terminal's network interface to a communications network node, without the terminal having access to information being communicated." Ex. 2003 ¶¶ 47-49.

The '969 Patent is directed to a portable device that may communicate with a remote networked device (*e.g.*, a server) by "tunneling" data through the access terminal. Ex. 1001 Title, 1:18-25, 2:39-51, 3:41-4:30, 18:12-14, Figs. 1, 5, 9, 10. "Tunneling" data in the context of the '969 Patent means that the terminal serves as a "conduit" or "bridge" for the TCAP to communicate with the server or other network devices, by sending and receiving data that is intended to be accessed only by the TCAP and/or the server, not the terminal. *Id.* at 4:60-64, 9:5-7, 12:67-13:4, 28:54-57, 27:28-28:15, Fig. 10 ("if data moving out of the TCAP and across the [access terminal] were captured at the [access terminal], such data would not be readable because the data was encrypted by the TCAP's processor").

The tunneling of the TCAP is not merely an exemplary embodiment. It is a core part of the invention, appearing not only in the Title, Abstract, and Field, but in the very name of the device—"tunneling client access point."—referenced repeatedly throughout the patent. Ex. 1001 Title, Abstract ("The TCAP "***tunnels" data through*** an access terminal's (AT) input/output facilities. . . . This enables the user to observe data stored on the TCAP ***without it being resident on the AT***, which can be useful to maintain higher levels of data security. Also, the TCAP may ***tunnel data through an AT*** across a communications network to access remote servers."), Field ("The present invention is directed … more particularly, to an apparatus, method and system to execute and process data by ***tunneling access through*** a terminal."), 1:11-14, 3:41-42, Fig. 2; *see Elbit Sys. Land and C4I Ltd. v. Hughes Net. Sys., LLC*, No. 2-15-cv-37-RWS-RSP, 2016 WL 6082571, at *5 (E.D. Tex. Oct. 18, 2016) (statements in Title, Abstract, Field and Background define the invention).

Petitioner recognized the importance of "tunneling" in its prior petitions on the related '047 and '703 Patents by stating that "encryption is of the essence to a tunneling client." Ex. 2072 (IPR2019-00416 (Paper 1)) at 6; Ex. 20732 (IPR2019-

00584 (Paper 1)) at 6.[2] For example, the ’969 Patent discloses that to prevent the terminal from accessing information sent between the TCAP and the remote server, both the TCAP and the server can include cryptographic server modules, which encrypt the data sent through the terminal. *Id*. at 19:38-20:24, 27:28-28:15, Figs. 9, 10. Thus, the terminal has no access to the data tunneled through it. Accordingly, when the claims refer to the portable device communicating with the network node “through” the terminal [network] interface, what it means is that the terminal has no access to information being communicated. Ex. 2003 ¶¶ 47-49.

D. <u>Node</u>

The term “node” appears in all independent Challenged Claims and thus all Challenged Claims. The Board should construe this term to have its plain and ordinary meaning. Petitioner cites a description in the ’969 specification of exemplary nodes, Petition 15, but the patent does not state that those examples are meant to be a definition. Ex. 2003 ¶ 50. Furthermore, the term “node” is widely

[2] As discussed in Exhibit 2003 ¶ 49, although tunneling does not require encryption, encryption is one way to accomplish what tunneling does require—that the terminal not have access to the tunneled information.

used in computer networks and a POSITA would readily understand it and would not require a definition. *Id.* ¶ 50.

VII. THERE IS NO REASONABLE LIKELIHOOD THAT PETITIONER WILL PREVAIL ON ANY CLAIM

A. <u>Ground 1: Iida Does Not Anticipate Claims 1-8, 10-16, 19-21, 24, 25, or 27-29</u>

As discussed below, Iida does not teach numerous elements of the challenged claims, and therefore cannot anticipate.

1. <u>Iida does not teach an "interactive user interface"</u>

All the Challenged Claims require an IUI. The Board, in a related proceeding recently acknowledged Iida's shortcomings with respect to the IUI elements:

> Based on the current record, we have concerns regarding whether Petitioner's arguments and evidence sufficiently show that Iida discloses the [IUI] limitations recited in claim 1 . . . . Although Iida discloses that '[t]he instruction inputted by the user is transmitted from the portable terminal 14 to the digital still camera 12,' . . . based on the current record, we are not sufficiently persuaded the user interacts with the menu on the display of the portable terminal. Instead, at this juncture, it appears the user is interacting with the numeric keyboard, which is not part of the display.

Ingenico Inc. v. IOENGINE, LLC, IPR2019-00416 (PTAB July 15, 2019), Paper 20 at 79 (citations omitted).

The Board's analysis is correct. All independent Challenged Claims require that the IUI is presented on the "terminal output component." And the "input component" on the terminal allows the user to interact directly with the interactive user interface that is presented on the terminal. These three elements are distinct requirements of each independent Challenged Claim. Petitioner's arguments ignore these clear requirements, seeking to blur them together avoid having to point to the required disclosures in Iida because Petitioner cannot do so. Petition 24-26.

Further, as discussed in section VI.A, this term should be construed to mean "a presentation containing interface elements with which a user may interact to result in the terminal taking action responsively by modifying what is presented." Thus, the device that presents the interface must respond to the user's input by modifying the presentation. Ex. 2003 ¶¶ 37-45. Iida fails to teach an IUI.

For this limitation, Petitioner cites the static menu image displayed on the display unit of Iida's portable apparatus. Petition 17-20, 24-26 (citing Ex. 1003 [0054], [0065], [0069], [0070], [0083], [0084], [0099]-[0114], [0131], Fig. 4C). But the screen of Iida is not an IUI as used in the '969 Patent, because (1) it presents no interface elements for the user to engage with, and (2) action responsive to user input is not taken by the portable apparatus that displays the image. *Id*. [0084]-[0085], [0098], Fig. 4A. Specifically, the portable apparatus merely displays a static menu image containing a message requesting that the user input numbers on a keypad. *Id.* [0068], [0083]-[0085], [0096], [0143], Figs. 6A-C, 6E-F, 6H-I; Ex. 2003 ¶ 52.

Moreover, the portable apparatus takes exactly the same action regardless of which number the user inputs—transmitting the number to the camera. It does not take action responsively, much less take action responsively to modify what is displayed. Ex. 1003 [0084]. It is the camera in Iida that transmits another fully formed static image to the portable apparatus for display. *Id*. [0083], [0085], [0089], [0096], [0099]; Ex. 2003 ¶ 53.

This is especially evident in Iida's treatment of "scrolling." Rather than true interactive "scrolling" of a display, Iida's apparatus merely "chang[es]-over" a static image on the display: "[W]here the instruction for changing-over (scrolling) the screen has been given by the user, . . . the image selection screen is generated [by the camera] using photographed image data of a plurality of other images (not yet displayed), and the generated information is transmitted to the portable terminal 14, thereby to change-over the image selection screen…" Ex. 1003 [0098]. Thus, Iida does not meet the "interactive user interface" limitation. Ex. 2003 ¶ 54.

2. **<u>Iida does not teach "communications . . . to a communications network node through the terminal [network] communication interface" or "communication[s] . . . through . . . the terminal network interface to a communications network node"</u>**

As discussed above in § VI.C, these limitations, which appear in independent Challenged Claims 1, 28, and 29, require "communications sent through the terminal's network interface to a communications network node, without the terminal having access to information being communicated."

Petitioner cites passages of Iida that generally describe communications between a portable apparatus and an image server 18. Petition 18-20, 22-24, 26-27 (citing Ex. 1003 [0014], [0069], [0070], [0110], [0113], [0114], [0131], Figs. 4C, 4D). But unlike the '969 Patent, Iida describes no measures to encrypt or secure communications between the camera and the server or to prevent the portable apparatus from having access to the images being sent. Ex. 2003 ¶ 56. Rather, the portable apparatus of Iida has access to the images, since the images are "transferred" to the portable apparatus and displayed on the screen of the apparatus. Ex. 1003 [0013] (images are saved to the server "in such a way that the image data is transferred to the communication apparatus"), [0015], [0029]-[0030] ("transfer component…transfers the image data to the image display apparatus…so that the image…may be displayed on the display unit."), [0039], [0040], [0043]-[0045].

In contrast, the invention described in the '969 Patent secures the information communicated with the network node through the terminal network interface in order to prevent the terminal from accessing it. Ex. 1001, 13:1-4 ("if data moving out of the TCAP and across the [access terminal] were captured at the [access terminal], such data would not be readable because the data was encrypted by the TCAP's processor."), 19:38-20:24 (describing cryptographic server module in the server), 27:28-28:15 (describing cryptographic server module in the TCAP), Figs. 9, 10; Ex. 2003 ¶ 57. This is a central purpose of "tunneling," which the Petition ignores.

Moreover, the '969 Patent makes clear that there is a distinction between a portable device (1) communicating ***"with"*** or ***"to"*** the terminal, meaning that the terminal has access to the information being communicated (which the terminal may forward along), and (2) communicating to a communications network node ***"through"*** the terminal's network interface, meaning that the information communicated is not accessible to the terminal. The Challenged Claims require both: "facilitate communications ***to*** the terminal and to a communications network node ***through*** the terminal [network] communication interface." Claims 1, 28, 29 (emphasis added). Petitioner's interpretation negates this difference.

Accordingly, Iida does not teach "communications . . . to a communications network node through the terminal [network] communication interface" or "communications . . . through . . . the terminal network communication interface to a communications network node" as required by independent claims 1, 24, and 27. Ex. 2003 ¶¶ 55-58.

3. **Iida does not teach "first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component"**

This limitation appears in all the Challenged Claims. As discussed in section VI.A, the Board in the above-referenced proceeding on the related '047 Patent has expressed doubt that Iida discloses an interactive user interface since "it appears the user is interacting with the numeric keyboard, which is not part of the display." IPR2019-00416, Paper 20 at 79.

Moreover, in Iida, every detail of the menu image displayed on the portable apparatus of Iida is dictated by the camera; the portable terminal is not involved in determining either the content or the arrangement of the images or text displayed. Ex. 1003 [0083], [0085], [0089], [0096], [0121], [0132] (describing how menu images are generated by the camera and transmitted to the portable apparatus).

Contrary to the Petitioner's theory, Petition 17-18, it is not sufficient for the control unit 60 of the portable apparatus to "execute[] a process . . . judg[ing] the received information to be information for displaying a screen on the display unit 62," and "display[] the menu screen on the display unit 62 by using the received information." Ex. 1003 [0083]. To present an IUI, the program code executed by the terminal processor would have to be involved in determining what is displayed, rather than simply displaying exactly what the portable device provides. Iida's portable apparatus merely displays whatever menu image the camera provides. *Id.*

Thus Iida does not anticipate claims 1, 28, and 29. Ex. 2003 ¶¶ 59-61.

4. **Iida does not teach executing fourth program code/fourth program code configured to be executed "in response to a communication received by the portable device resulting from user interaction with the interactive user interface"**

This limitation appears in all the Challenged Claims. Even if Iida disclosed an interactive user interface, which it does not, as discussed above, § VII.A.1, it certainly would not disclose execution of program code "in response to a communication received by the portable device resulting from user interaction ***with*** the interactive user interface."

The user in Iida merely pushes a button on a numeric keypad which, as acknowledged by the Board in the above-referenced, related case involving the '047 Patent, is not part of an IUI, but rather the input component for the portable apparatus. Ex. 1003 [0068], [0084]; IPR2019-00416, Paper 20 at 79. In contrast, the '969 Patent describes the user as directly "engaging" with "interface elements." Ex. 1001 at 1:52-56, 10:32 ("Should the user engage a user interface element…"), 10:37-38 ("engaging the appropriate user interface element…"), 11:13-15, 61-64. It is not sufficient to disclose just any communication to the portable device, it must be a communication that "is received by the portable device resulting from user interaction ***with*** the interactive user interface." Thus, since the fourth program code limitations require the code to be configured for execution in response to user interaction ***with*** the IUI, Iida does not disclose them. Ex. 2003 ¶¶ 62-63.

5. <u>**Iida does not teach "program code which, when executed by the terminal processor, is configured to provide a communications node on the terminal" or "program code [which, when executed by the portable device processor, is configured]/[stored on the portable device memory] to provide a communications node on the portable device"**</u>

These limitations appear in all the Challenged Claims. Claims 1 and 28 further require the communications node to "facilitate communications to the portable device ***and*** to a communications network node through the terminal network communication interface." In independent Challenged Claim 29, the portable device is configured "to facilitate communications to the terminal ***and*** to a communications network node through the terminal communication interface."

Iida is silent regarding any program code that provides a communications node on either the portable apparatus or the camera, or, as Petitioner contends, "establish[es] terminal 14 as a node" or "establish[es] the camera as a node." Petition 18-20, 22-24 (citing Ex. 1003 [0014], [0045], [0066], [0068]-[0070], [0083], [0084], [0109], [0110], [0113], [0114], [0144]). Petitioner and its expert Mr. Geier speculate that "a POSITA would have understood" or "would understand" that "control unit 60 had executed code to establish the terminal as a node," and "the camera necessarily includes code in ROM or RAM for establishing the camera as a node." Petition 20, 24; Ex. 1002 ¶¶ 53-54. But they identify nothing in Iida that refers to such program codes. Ex. 2003 ¶ 64.

The Petition is ambiguous as to whether the "communications node on the terminal" is purportedly met by Iida's portable apparatus 14, the second wireless communication unit 68, or the first wireless communication unit 66. Petition 18-19. Yet, none of these components meets the requirements of the "communications node on the terminal." The portable apparatus is already cited as the "terminal," Petition 16, so it cannot also be the "the communications node ***on*** the terminal." Neither the second wireless communication unit, nor the first wireless communication unit can be the communications node, because neither is involved in communication with both the camera (Petitioner's alleged "portable device") ***and*** the server 18 (Petitioner's alleged "communications network node"). Specifically, the second wireless communication unit communicates with the camera but not the server, Ex. 1003 [0083]-[0084], whereas the first wireless communication unit communicates with the telephone network but not the camera, *id*. [0069]. Thus, Petitioner has identified no structure in Iida which meets the requirements of the claimed "communications node on the terminal."

Petitioner similarly conflates the claimed "portable device" with the "communications node ***on*** the portable device," contending that Iida's camera ***is*** a "portable device," Petition 13, while simultaneously contending that the camera is a "communications node ***on*** the portable device." Petition 22-24. By failing to identify both elements within Iida, Petitioner has failed to meet its burden with respect to the "communications node on the portable device." Therefore, Iida fails to teach these additional program code limitations and cannot anticipate claims 1, 28, and 29. Ex. 2003 ¶¶ 64-65.

6. **The challenged dependent claims are not anticipated by Iida**

Iida does not anticipate the dependent Challenged Claims for at least the reasons discussed above regarding the independent Challenged Claims, and the following additional reasons.

a. **Claim 3**

Claim 3 requires that the communication transmitted to the communications network node "facilitates verification of the ***portable device***." (emphasis added). The Petition applies this limitation to the ***user***-authentication feature of the Iida system, mischaracterizing it as "the same process described in the '969 Patent for verification." Petition 29 (citing Ex. 1003, [0075], [0113]). But the camera in Iida is a rental device which "is used conjointly by a plurality of users." Ex. 1003 [0125]. Thus, the user ID is associated only with the current user, not with the camera. Ex. 1003, [0075] ("When the user has paid the necessary fee, the shop affords **a user**

ID to the user . . . .") (emphasis added).[3] Accordingly, the user ID in Iida cannot be used to verify the portable device. Ex. 2003 ¶ 66.

b. **Claim 4**

Claim 4 specifies that the communication caused to be transmitted to the communication network node "facilitates the transmission of encrypted communications from the communication network node to the terminal." Rather than identifying anything relevant in Iida, Petitioner contends that the "encrypted communications" element should be denied patentable weight under the "printed matter doctrine." Petition 30-32. But Petitioner stretches the printed matter doctrine beyond its boundaries, including those established by Petitioner's own cases.

"Differences between an invention and the prior art cited against it cannot be ignored merely because those differences reside in the content of the printed matter." *In re Gulack*, 703 F.2d 1381, 1385 (Fed. Cir. 1983). It is only if the "printed matter is not functionally related to the substrate [i.e., the medium on which it resides]" that it is not given patentable weight. *Id.*; *Praxair Distribution Inc. v. Mallinckrodt Hospital Prods. IP Ltd.*, 890 F.3d 1024, 1031-32 (Fed. Cir. 2018).

[3] The '969 Patent discusses portable device verification separately from user verification. *See* Ex. 1001 7:12-24, 7:64-8:13, 22:40-42, 24:58-62.

Petitioner's argument that the encrypted communications in claim 4 "serves no function" (Petition 32) disregards the functional purpose of encryption, which is to block access to the encrypted information. Indeed, the case law cited by Petitioner cautions against applying the printed matter doctrine to information that is processed by a computer, such as the claimed encrypted communications. *See In re Lowry*, 32 F.3d 1579, 1583 (Fed. Cir. 1994) ("The printed matter cases have no factual relevance where…the information [is] processed…by a machine, the computer."). Petitioner has identified no case to the contrary. Petition 37 (citing *Lowry*, 32 F.3d at 1583; *In re Nuijten*, 500 F.3d 1346, 1365 (Fed. Cir. 2007) (Linn, J., concurring and dissenting); *Praxair*, 890 F.3d 1024 (claim terms unrelated to information processed by a computer)).

The Petition also points out that ***a storage medium*** (*e.g.*, a memory stick) containing encrypted information representing the number of times remaining for the user to save images can be plugged into Iida's camera to copy that information. Petition 32 (citing Ex. 1003 [0150]). But this does not constitute transmitting or receiving encrypted ***communications***, much less encrypted ***communications from a communication network node*** as required by claim 4.

Petitioner misapplies *Catalina Mktg. Int'l, Inc. v. Coolsavings.com, Inc.*, 289 F.3d 801 (Fed. Cir. 2002). Petition 32. Under *Catalina*, the context of the claim language is important. For example, the same term might be limiting in the body of a claim, but not in the preamble. *Catalina*, 289 F.3d at 808−11. . Furthermore, in the present case, not only is the term in question in the body of the claim, but unlike in *Catalina*, it relates to a feature (encrypted communications) with the specific technical function of securing information transmitted between, and used by, two machines. Taken in context, the limitation should be considered limiting.

Thus, the limitations relating to transmitting encrypted communications in claim 4 must be given patentable weight and they are not taught by Iida. Ex. 2003 ¶ 67.

c. **<u>Claims 5-11</u>**

Claims 5-11 recite, among other things, that the communication network node comprises a database. A POSITA would understand that a database is more than just a collection of files. It is an ***organized*** collection of data. Ex. 2003 ¶ 68. Non-limiting examples include sets of records that are organized by ordering or creating relations between them. *Id.* The "image data saving areas" of Iida on which Petitioner relies, Petition 33, do not qualify as databases because there is no evidence that they are organized in any way. Ex. 1003 [0070], Fig. 1C.

d. **<u>Claim 7</u>**

Claim 7 specifies that the communication caused to be transmitted to the communication network node "facilitates the download of program code on the communication network node to the terminal." Since Iida does not teach downloading program code, Petitioner again misapplies the printed matter doctrine, contending that the "program code" limitation should be denied patentable weight. Petition 34. But as discussed above, it is only if the "printed matter is not functionally related to the substrate [i.e., the medium on which it resides]" that it is not given patentable weight. *In re Gulack*, 703 F.2d at 1385; *Praxair*, 890 F.3d at 1031-32.

Petitioner's contention that the program code in claim 7 "provides no function" (Petition 34) disregards what a computer program is. The sole purpose of program code is to impart functionality to the apparatus that runs it. Furthermore, the specification of the '969 Patent explains that the purpose of receiving the program code is to update the device's software. *See* Ex. 1001 at 9:32-37, 19:15-17 (referring to TCAP obtaining program updates from the server). A software update is functionally related to the device receiving it. Thus, the limitations relating to downloading program code in claim 7 must be given patentable weight and are not taught by Iida.

e. **Claim 10**

Claim 10 specifies that the communication transmitted to the communication network node "facilitates synchronizing content on the portable device with content on the communication network node database." The data synchronization described in the '969 Patent involves synchronizing the contents of the portable device and server, including receiving updated data and program code (*e.g.*, from a server) and overwriting old data and program code in the portable device memory with the updated data, or *vice versa*. Ex. 1001 at 8:28-31; 9:32-37; Ex. 2003 ¶ 70.

In contrast, Petitioner and its expert Mr. Geier contend that any pair of devices with the capability of storing data must "facilitate synchronizing data" as in claim 10, because a user could take the initiative to save images of his or her own choosing. Petition 35; Ex. 1002 ¶ 70. But that is hindsight speculation based on the teachings of the '969 Patent. Iida says nothing about synchronization.

f. **Claims 11 and 12**

Claims 11 and 12 require that the communication transmitted to the communication network node "facilitates the download of a live data feed to the terminal." Since Iida says nothing about a live data feed, Petitioner disregards this claim term, arguing that "[w]hether the data being downloaded is coming from storage or being produced live is of no consequence," and the mere concept of "facilitat[ing] a download" is sufficient to teach downloading a live data feed. Petition 35. But there is no evidence that the portable apparatus of Iida is capable of receiving anything other than individual image files, or that there would be any reason for someone to view or listen to live data feeds on Iida's portable apparatus.

In addition, there is no evidence that the Iida system has enough bandwidth to accommodate live feeds. The only data transmitted in Iida are discrete image data files, which are not received live, but downloaded in their entirety before being stored or viewed. Ex. 1003, [0106] (entire image transferred to user's PC and saved); [0114] (full image transferred to server, then control unit 52 awaits acknowledgment); [0128] (camera generates image selection screen and transmits the full information to the terminal for display). Thus, Iida does not teach the additional limitations of claims 11 and 12. Ex. 2003 ¶ 71.

Petitioner's reliance on *Catalina* and *Ex Parte Haworth, et al.*, No. 2009-000350, 2009 WL 2342033 (BPAI July 30, 2009), Petition 36, is misplaced. As discussed above, in *Catalina*, the context of the claim language was important. In the present case, the term in question is in the body of the claim, and also relates to a feature with the specific technical function of providing a live data feed to be received and used by a machine. In this context, the limitation should be considered limiting. *Haworth* appears to have involved improperly drafted means plus function claims. 2009 WL 2342033. Neither case involved claims analogous to claim 12 of the '969 Patent.

The "live data feed" limitation of claim 12 should be given weight, and it is not taught by Iida.

g. <u>**Claims 13-16**</u>

Claims 13-16 specify that the portable device is "further configured to affect the presentation of the interactive user interface on the terminal output device." As discussed above in Section VII.A.1, Iida does not teach an interactive user interface. Therefore, it cannot anticipate claims 13-16. Claim 16 contains the additional limitation that the IUI is affected to present a user name and/or a portable device identifier. The Petition tries to negate this limitation under the "printed matter doctrine." Petition 38 (citing *Lowry*, 32 F.3d at 1583; *Nuijten*, 500 F.3d at 1365; *Praxair*, 890 F.3d 1024). As discussed above, the printed matter doctrine does not apply to information which is processed by a computer or machine. *See, e.g.*, *Lowry*, 32 F.3d at 1583. Thus, the user name and/or portable device identifier of claim 16 must be given patentable weight, since it is stored on the portable device memory and presented on the terminal output component.

Petitioner misleadingly quotes the text "SINCE THE CURRENT REMAINING NUMBER OF TIMES IS ZERO, THE CAMERA CANNOT BE USED" in the display of Figure 6B as simply "the camera" to try to argue that it represents a portable device identifier. Petition 38. Petitioner's speculation that the above text "could be written to be more or less specific about the camera or its user," *id.*, is not based on anything in Iida.

h. **<u>Claims 19 and 20</u>**

Claims 19 and 20 specify that the portable device is configured to cause the terminal to present an IUI on the terminal output component. As discussed above in

Section VII.A, Iida does not disclose an IUI, and it is the camera, not the portable apparatus, that presents the interface in Iida.

i. **Claim 21**

Claim 21 specifies that the IUI comprises a GUI. As discussed above, the proper definition of a GUI is: "a presentation containing graphical interface elements with which a user may interact to result in the terminal taking action responsively by modifying what is presented."

Petitioner contends that Figure 6E of Iida shows a GUI because (1) it includes static thumbnail images, and (2) according to Petitioner's expert Mr. Geier, "a touch pad was known to manipulate a cursor on graphic displays such as shown in Fig. 6E for making a selection of one of the displayed images." Petition 40−41; Ex. 1002 ¶ 77. But the user does not click, tap, use a cursor on, or otherwise interact with the thumbnails in Iida, but merely looks at them and presses a numeric key separate from the display. Ex. 1003, [0068], [0095]-[0097], [0128]. There is no connection between the "touch pad" in paragraph [0068] and the display illustrated in Fig. 6E, since the only way that the user can react to the display is to "input [the] number affixed to the image" that (s)he wants to view. Ex. 1003, [0097], Fig. 6E. Nothing in Iida suggests using a touch pad to enter a number.

In addition, as discussed above with respect to claim construction, *supra* § VI.B, a GUI, like any IUI, requires the terminal to take action responsively to user interaction by modifying what is presented. Iida's portable apparatus does not. Rather, it takes the same action regardless of which number the user inputs—transmitting the number to the camera. Ex. 1003 [0084].

Thus, Iida does not meet the "graphic user interface" limitation of claim 21. Ex. 2003 ¶¶ 74-75.

B. **Ground 2: Claims 1-8, 10-16, 19-21, and 24-29 are Not Obvious Over Iida and the Fuji Guide**

As discussed above, *supra* § V.B, Petitioner has not proven that the Fuji Guide is prior art. Petitioner and its expert also do not identify in the Fuji Guide at least the limitations of independent Challenged Claims 1, 28, and 29 discussed in Sections VII.A.1-5 above.

Among other things, the Fuji Guide does not refer to any program code stored on or executed by the camera, which Petitioner contends is the "portable device" of the claims. Instead, the only software described in the Fuji Guide runs on a PC, which Petitioner contends is the "terminal" of the claims. Ex, 1004, *passim*; Petition 54-61.

Petitioner contends that the FinePixViewer startup window meets the limitation "first program code which, when executed by the terminal processor, is configured to present an interactive user interface on the terminal output component," of the independent Challenged Claims. Petition 57-58. But claims 28 and 29 require that the ***portable device cause*** (claim 28) or ***be configured to cause*** (claim 29) the terminal to execute the first program code. *See also* dependent claims 19 and 20. The Petition cites nothing in the Fuji Guide to suggest that the camera takes any action to cause the FinePixViewer startup window to appear. Rather, the only device involved in displaying that window is the PC, which Petitioner contends is the claimed "terminal." For this additional reason, the Fuji Guide does not fill-in Iida's holes, and Iida and the Fuji Guide, even if combined, could not render obvious Claims 28 and 29.

Likewise, Petitioner does not explain how Iida and the Fuji Guide software would work together to implement fourth program code that is executed or configured to be executed ***by the portable device*** (which Petitioner contends is the camera in each of these references) "in response to a communication received by the portable device resulting from user interaction with the ***interactive user interface***" as required by all the independent Challenged Claims—much less such program code which, when executed by the portable device processor, is configured to cause a communication to be transmitted to the communication network node" as required by claim 2. Iida discloses no IUI and thus cannot disclose program code executed (or configured to be executed) in response to a communication resulting from user interaction with an IUI. And the Fuji Guide does not disclose any program code executed by the camera. Thus, Iida and the Fuji Guide cannot render any of the Challenged Claims obvious even if combined.

In addition, a POSITA would not have been motivated to combine Iida and the Fuji Guide. It would be technically challenging to adapt the software described in the Fuji Guide to install and run on Iida's portable apparatus because the technical requirements of the Fuji software are substantially greater than what would have been available to a portable apparatus such as a PDA. The Fuji software requires a desktop operating system, a 200 MHZ Pentium Processor, 64 MB of RAM, 300 MB of hard drive space, and an 800 x 600 pixel or better display with at least 16 bits of color or better. Ex. 1004 at 8. These requirements are far beyond what would have been available in a PDA at the time, and certainly beyond what Iida contemplates. Ex. 2003 ¶ 79. Given the differences in the technical requirements, a POSITA trying to combine Iida and the Fuji Guide would have to re-architect the Fuji software code. *Id*.

Combining Iida and the Fuji Guide also would not function without material modification. As discussed above, the menu screen in Iida is generated by the camera, not the portable apparatus. Ex. 1003 [0083], [0085], [0089], [0096], [0121], [0132]. In contrast, the FinePixViewer startup window in the Fuji Guide is generated by software running on the PC. Ex. 1004 at 17. In order to combine the menu screen of Iida and the FinePixViewer startup window of the Fuji Guide, a

POSITA would need to decide which device generates the menu and then reprogram the software accordingly. There is no indication in Iida or the Fuji Guide as to how to make this choice or how this should be accomplished. Ex. 2003 ¶ 80.

It also would make no sense in the Iida system, where the camera performs all of the processing, to employ the interface of the Fuji Guide, where the PC performs all of the processing. Ex. 1004, *passim*. Nor would there be any reason for the Fuji interface to communicate with the camera in the manner of Iida, because the images are stored in and viewable on the PC using the Fuji Guide interface. Ex. 1004, 17-18, 37-38. There is also no need for the camera in the Fuji Guide to communicate with an outside network like that contemplated by Iida, because the PC in the Fuji Guide handles all communications with the server. Ex. 1004, *passim*; Ex. 2003 ¶ 81.

The Petition cites Iida's mention of the FinePix6800z camera, Petition 54, but fails to acknowledge that Iida contains no mention of the Fuji Guide. Ex. 1003 [0153]. The Fuji Guide describes software to be installed on the home PC of a user who owns, not rents, a camera. Ex. 1004, 3 ("Thank you for ***purchasing*** these FUJIFILM products.") (emphasis added), 9 (referring to "the particular camera model you ***purchased***") (emphasis added); Ex. 1005 ¶ 3 (Petitioner's witness Mr. Widener purchased, not rented, his camera). The purpose of Iida, in contrast, is to provide a lower-cost, ***rental*** alternative to more expensive digital cameras at the time that included on-board LCDs. *See supra* § V.A. Even assuming that the Fuji Guide were provided with the ***purchase*** of a camera as Mr. Widener alleges, there is no evidence that the Guide or the software described in it would have been provided to a camera ***renter*** as contemplated by Iida. To the contrary, a typical user in 2004 would have no interest in installing a suite of new software on his/her home computer in order to rent a camera temporarily. Ex. 2003 ¶ 82. Thus, the systems described in the two references are for different purposes. *E.g.*, *Redline Detection, LLC v. Star Envirotech, Inc*., 811 F.3d 435, 453 (Fed. Cir. 2015).

Indeed, a key feature of Iida is restricting how many times the renter can take a photo and save the image, which is controlled through a value called the "number of times," representing the remaining number of uses that the renter has paid for. Ex. 1003 [0013], [0033]-[0034], [0050], [0085]-[0090]. The software described in the Fuji Guide is unsuitable for this purpose because it allows the user to save images on his/her home PC without restriction, thus defeating the usage-control feature of the Iida system. Ex. 1004, 18.

Next, Iida contemplates portability as a goal, consistently referring to the device connected to the camera as a "portable" apparatus such as a portable telephone, PDA, wearable computer, or mobile computer. Ex. 1003 *passim*. In contrast, there is nothing in the Fuji Guide to suggest that the PC is portable. Thus, the combination would not work for Iida's intended purpose of portability.

Finally, Petitioner does not attempt to identify anything in the Fuji Guide that might be relevant to the additional limitations of dependent Challenged Claims 2-6, 8, 10, 13-16, 19-21, 24, 25, or 27. Petition 54, 56.

Regarding claims 11 and 12, Petitioner contends that it would be obvious to add the videoconferencing capability of the software in the Fuji Guide to Iida's portable apparatus. Petition 62-63. But the Petition identifies nothing in either reference to suggest that a person renting a camera and using it with a portable apparatus such as a PDA would be motivated to use those devices for videoconferencing as described in the Fuji Guide.

Regarding the USB interface in claim 26, Iida mentions wired connections only to say that they are ***disfavored***. Ex. 1003 [0014] (Communication between the devices "should preferably be wireless communication such as the Bluetooth when the labor of connection a communication cable, etc. are considered.") Thus, a POSITA would not be motivated to seek additional ways to make a wired connection, such as USB. Ex. 2003 ¶ 87.

The Petition identifies no technical shortcoming or problem expressed in either Iida or the Fuji Guide that would be solved by combining them, or any way that the systems would be improved by combining them. *See, e.g., Cross Med. Prods., Inc. v. Medtronic Sofamor Danek, Inc.*, 424 F.3d 1293, 1321-23 (Fed. Cir. 2005). For example, the Petition refers to Iida's mention of the Fuji camera, Petition

54, but fails to identify any passage in Iida that suggests a need for ***a rental camera user*** to install special software on his/her home computer. To the contrary, the combination would be undesirable, since a renter would not want his/her computer cluttered with software that would become useless as soon as the camera is returned to the rental shop. Ex. 2003 ¶ 88.

Thus, a POSITA would not have considered it obvious to combine Iida and the Fuji Guide, much less to do so to arrive at any Challenged Claim.

C. <u>Ground 3: Claim 4 Is Not Obvious Over Iida in View of Shaffer</u>

In their discussion of Ground 3, Petitioner and its expert do not identify in Shaffer any of the foregoing limitations of independent Challenged Claim 1 which, as discussed above with respect to Ground 1, are missing from Iida. For at least that reason, Petitioner fails to meet its burden with respect to Ground 3.

In addition, a POSITA would not have been motivated to combine Iida and Shaffer. First, the systems described in the two references are for different purposes. *E.g.*, *Redline*, 811 F.3d at 453. As discussed above, the system in Iida is for providing a lower-cost, rental alternative to more expensive ***digital*** cameras at the time that included on-board LCDs, whereas the Shaffer system is for processing and printing images from a ***film*** camera. *See supra* § V.A, V.C. Although Iida describes a remote photo lab that produces prints based on digital images from the camera, Ex. 1003 Abstract, [0004], [0033]-[0036], [0072], [0076], [0138], there is no indication that Iida's photo lab is capable of processing film. Moreover, because Iida's camera is digital, it would make no sense to combine it with Shaffer's film-scanning system.

In addition, claim 4 refers to encrypted communications, yet Petitioner cites nothing in Iida to suggest a concern about the security of camera images that might motivate a POSITA to incorporate the concepts of Shaffer. The only information that Iida considers sensitive enough to encrypt is the remaining number of times that the user is allowed to use the camera. Ex. 1003 [0150]; Ex. 2003 ¶ 91. Petitioner and its expert Mr. Geier contend that (1) "many customers prefer that the photographs or the ordering information not be disclosed to the public"; and (2) "[a] POSITA, having Iida in hand, would have been motivated to look to Shaffer as Shaffer describes methodologies for secure transmission of image data over the Internet." Petition 65-66; Ex. 1002 ¶ 105. But neither Petitioner nor Mr. Geier identify any teaching ***in Iida*** that would inspire their hypothetical POSITA with "Iida in hand" to seek ways to secure the transmission of images.

In addition, Shaffer encrypts a high-resolution image or the difference between a high-resolution image and its low-resolution counterpart, used to receive or reconstruct a high-resolution image at the customer site. Ex. 1006, 5:46-47, 6:6-18. There is no suggestion in Iida of a need to create different high- and low-resolution versions of an image. Nor would there be any reason to receive or reconstruct a high-resolution image within the portable apparatus of Iida, because

PDAs in the relevant time period (ca. 2004) did not commonly have screens with sufficient resolution to view a high-resolution image. Ex. 2003 ¶ 92. Indeed, Iida contemplated only a "display quality" resolution of 100,000 (equivalent to 316 x 316) pixels, which was sufficient for "displaying the thumbnail image and checking the content thereof," but would not have been sufficient to view the 3072 x 2048 pixel, high-resolution images of Shaffer. Ex. 1003 [0100]; Ex. 1006, 3:17-20; Ex. 2003 ¶ 92. Thus, for the portable apparatus of Iida to receive the encrypted high-resolution data of Shaffer would serve no purpose. Ex. 2003 ¶ 92.

Even if one were to combine Iida and Shaffer, Petitioner has failed to demonstrate that the combination contains the elements of claim 4. First, claim 4 depends from claim 2, including fourth program code, which, when executed by the portable device processor, is configured to cause a communication to be transmitted to the communications network node. According to underlying independent claim 1, the fourth program code is configured to be executed in response to a communication received by the portable device ***resulting from user interaction with an IUI*** presented on the terminal. Shaffer says nothing about what kind of interface the disclosed system might use. Thus, it cannot teach the fourth program code required by claim 4 through its dependence from claims 1 and 2.

In addition, Shaffer's system has only one customer device, personal computer 32. Ex. 1006, Fig. 1. There is no mention of a portable device anywhere in Shaffer. For claim 4, sending encrypted images is not enough. Shaffer would have to teach, among other things, (1) executing program code by ***a portable device processor*** to cause a communication to be transmitted to a communications network node, and (2) transmission of encrypted communications from the communications network node to a ***terminal***—*i.e.*, a component other than the portable device. Shaffer cannot, because it describes only a single customer device. Ex. 1006, Fig. 1. Accordingly, even if Shaffer were combined with Iida, the combination would not satisfy the limitations of claim 4.

The Petition also cites Ford for the proposition that "encryption techniques were well known." Petition 64, 66. But Petitioner and Mr. Geier identify nothing in Iida or Shaffer that would motivate a POSITA to combine them with the encryption methods of Ford. Petition 66; Ex. 1002 ¶ 106.

Thus, Petitioner has failed to meet its burden for Ground 3.

D. Ground 4: Claim 4 Is Not Obvious Over Iida and the Fuji Guide in View of Shaffer

As discussed above with respect to Ground 1, Iida fails to teach, the limitations of the independent Challenged Claim 1 discussed above in Sections VII.A.1-5. In Ground 4, Petitioner fails to identify any portions of the Fuji Guide or Shaffer that remedy those shortcomings of Iida. Petition 67-68.

In addition, for the reasons discussed above with respect to Grounds 2 and 3, a POSITA would not have considered it obvious to combine Iida with either the Fuji Guide or Shaffer. It also would not have been obvious to combine the Fuji Guide with Shaffer.

The software described in the Fuji Guide allows a user to save and manage photos from a ***digital*** camera. Ex. 1004, at 7. In contrast, as discussed above with respect to Ground 3, *supra* § VII.C, the Shaffer system is for processing and printing images from a ***film*** camera. The Fuji Guide makes no mention of a photo lab for processing film, nor would that make sense, since the camera in the Fuji Guide is digital. Ex. 2003 ¶ 97.

There is also no reason provided in Iida, the Fuji Guide, or Shaffer to combine them, and the Petition identifies none. In particular, claim 4 refers to encrypted communications, yet Petitioner cites nothing in either Iida or the Fuji Guide to suggest a concern about the security of camera images.

Even if one were to combine the systems of Iida, Fuji Guide and Shaffer, the combination would fail to meet the limitations of claim 4. For example, as discussed above with respect to Ground 3, Shaffer describes only a single, non-portable customer device, Ex. 1006 Fig. 1, and thus cannot teach (1) executing program code by a ***portable device processor*** to cause a communication to be transmitted to a communications network node, and (2) transmission of encrypted communications from the communications network node to a ***terminal***—*i.e.*, a component other than the portable device. The Petition identifies nothing in Iida or the Fuji Guide that remedies this shortcoming of Shaffer. Thus, Petitioner has failed to meet its burden with respect to Ground 4.

VIII. THE BOARD SHOULD EXERCISE ITS DISCRETION TO DENY THE PETITION UNDER 35 U.S.C. § 314(a)

The Board should deny the Petition because institution would be inefficient and wasteful of the Board's and the parties' resources. ***First***, after Petitioner injected itself into the district court proceedings and delayed for months, Petitioner and its indemnitee, PayPal, seek to unreasonably and vexatiously multiply the proceedings by filing a staged series of ***three*** separate IPR petitions on the '969 Patent, and altogether ***twelve*** IPR petitions on the three patents at issue in the district court. ***Second***, Petitioner's delay in filing the Petition until just two weeks before its indemnitee's one-year statutory bar date[4]—intended to give Petitioner's parent

[4] Although Petitioner's indemnitee, PayPal, was not served with the complaint until April 10, 2018, the complaint was filed and sent to PayPal on March 23, 2018. Ex. 2069. Thus, this Petition (and nine others by Petitioner or its indemnitee) were filed more than a year after Petitioner's indemnitee was on notice of IOENGINE's infringement allegations.

company an advantage in the district court—will now result in a wasteful race to judgment between the Board and the district court.

Ingenico's Petition represents prejudicial gamesmanship and wasteful inefficiency, and thus negatively impacts "both the efficiency of the *inter partes* review process and the fundamental fairness of the process for all parties." *Gen. Plastic Indus. Co. Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, slip op. at 18 (PTAB Sep. 6, 2017) (Paper 19) (precedential).

A. Petitioner and its Indemnitee Sequentially Filed Three Petitions Challenging the '969 Patent and Twelve Petitions Challenging Patent Owner's Patents

Months after IOENGINE filed the PayPal Action, Petitioner injected itself into the district court proceedings by filing the Ingenico Action. *See supra* § II. The district court proceedings are now coordinated for fact and expert discovery and claim construction, and share all dates until trial, to maximize procedural efficiency and minimize burden on the parties and court. Ex. 2031.

Petitioner's interests are closely aligned with those of PayPal, and Petitioner should have identified PayPal as a real party in interest.[5] Indeed, Petitioner acknowledged in its mandatory notices that it has an indemnification agreement with PayPal (Paper 6 at 1), and its declaratory judgment complaint relies on PayPal's indemnification request for purposes of standing. Ex. 1019 ¶ 9. Petitioner has further acknowledged that PayPal is its customer, and that certain products accused of infringement in the PayPal Action are "supplied to PayPal by Ingenico." Ex. 1019 ¶¶ 7-8, 10. IOENGINE has asserted infringement claims against both Ingenico and PayPal in the district court. Exs. 2029, 2030. Under these circumstances, the relationship between Petitioner and PayPal "incentivizes both parties to invalidate claims of [IOENGINE's asserted patents]" and "[i]n that sense, [PayPal] is a clear beneficiary of [Petitioner's] efforts in this *inter partes* review, and it follows readily

[5] This is important at least "to ensure that third parties who have sufficiently close relationships with IPR petitioners would be bound by the outcome of instituted IPRs under § 315(e), the related IPR estoppel provision." *See Ventex Co., Ltd. v. Columbia Sportswear N. Am., Inc.*, Case IPR2017-00651, slip op. at 6 (PTAB Jan. 24, 2019) (Paper 148) (precedential).

that [Petitioner] represents [PayPal's] interests in this proceeding." *See Ventex*, IPR2017-00651, slip op. at 8.

In its declaratory judgment complaint, Petitioner studiously avoided seeking a judgment of invalidity, indicating that it had already formulated its strategy of staged IPR challenges as early as June 2018. *See* Ex. 1019. More than a year after the PayPal Action was filed, nine months after filing its own declaratory judgment action, and three months after filing its first IPR petition in this set of proceedings, Ingenico filed this Petition challenging the '969 Patent (based on the same primary reference, Iida, as its earlier petitions). Shortly thereafter, Petitioner's indemnitee, PayPal, filed two more petitions challenging the '969 Patent. Where two petitioners are codefendants accused of infringing the same patents based on the same products, their decision to file sequential petitions weighs in favor of discretionary denial. *Valve Corp. v. Elec. Scripting Prods., Inc.*, IPR2019-00062, slip op. at 9-10 (PTAB Apr. 2, 2019) (Paper 13) (precedential).

In addition to this Petition, Petitioner has filed three additional IPR petitions challenging the related '047 and '703 Patents (all of which rely on Iida as the primary reference). Finally, Petitioner raised invalidity in the district court as well in its answer to IOENGINE's counterclaim, and served invalidity contentions that incorporate by reference all the IPR petitions filed by Petitioner and its indemnitee and all the art cited therein. Ex. 2035 at 112-13; Ex. 2036 at 6-7.

This Petition should thus be considered in its proper context alongside the other two petitions against the ’969 Patent and the total of eleven other petitions in Petitioner and its indemnitee’s campaign, and should be denied.

B. Petitioner Strategically Delayed Filing This Petition to Help its Parent Company in the District Court Proceedings

Petitioner timed this Petition to its strategic benefit in the district court, which is improper. *See Valve*, IPR2019-00062, slip op. at 13-14 (rejecting excuse for delay); *R.J. Reynolds Vapor Co. v. Fontem Holdings 1 B.V.*, IPR2017-01319, slip op. at 11 (PTAB Nov. 8, 2017) (Paper 7) (Board considers “non-strategic reasons” for delay). IOENGINE asserted infringement counterclaims against Petitioner’s French parent (Ingenico Group S.A.). Ingenico Group S.A. challenged personal jurisdiction by motion to dismiss. Ex. 2032. An issue central to the motion was the extent of the French parent’s contacts with the U.S. For Petitioner to name its parent as a real party in interest would be relevant to this analysis.

In an act of procedural gamesmanship, however, Petitioner waited to file its petitions until after the December 17, 2018 hearing in the district court on its motion to dismiss its foreign parent company. Ex. 2033. In fact, the first petition was filed later that day, after the motion was submitted to the district court, and named Ingenico Group S.A. as a real party in interest. In yet another example of procedural gamesmanship, after the district court ordered jurisdictional discovery, Petitioner withdrew its motion to dismiss just days before the deadline to respond to IOENGINE's jurisdictional discovery requests. Ex. 2046.

After filing its first petition on December 17, 2018 and another approximately a month later, Petitioner then waited over two months (until nearly the eve of its indemnitee's one-year statutory bar date) to file two additional petitions, including this Petition. In the interim, IOENGINE served its initial infringement charts on Petitioner, which, pursuant to the district court scheduling order, Ex. 2031 at 3-4, narrowed the number of claims IOENGINE was asserting. Ingenico's delay in filing this Petition thus gave it the benefit of learning which asserted claims IOENGINE was choosing to focus on in the litigation.

As explained below, the strategic delay of the Petition will, if instituted, result in a highly inefficient and prejudicial process involving overlapping claim construction proceedings and a race to judgment, as a final written decision would be expected after both district court trials.

C. The *General Plastic* Factors

The Board has recognized a set of non-exclusive factors to be considered in exercising its discretion to deny IPR petitions. *Gen. Plastic*, IPR2016-01357, slip op. at 15-16; Trial Practice Guide Update at 23-26 (July 2019) ("TPG Update").

The Board has emphasized that the *General Plastic* factors are not exhaustive, and "additional factors may arise in other cases for consideration, where appropriate." IPR2016-01367, slip op. at 18; TPG Update at 25-26. Although the *General Plastic* factors focus on follow-on petitions, "[t]here may be other reasons besides the 'follow-on' petition context" in which discretionary denial is appropriate. TPG Update at 25. For example, the Board has also considered the advanced stage of a pending district court proceeding, along with the similarities between that proceeding and the Petition, *see* TPG Update at 25-26; *Thermo Fisher Sci., Inc. v. The Regents of the Univ. of Cal.*, IPR2018-01367, slip op. at 21 (PTAB Feb. 7, 2019) (Paper 10); *NetApp, Inc. v. Realtime Data LLC*, IPR2017-01195, slip op. at 12-13 (PTAB Oct. 12, 2017) (Paper 9); and the merits of the petition, *see* TPG Update at 25.

Congress has taken note of the problem of "abusive serial petitions…either by the same petitioner or different petitioners," and expressed concern that the "*General Plastic* factors only are not sufficient." Ex. 2045 at 1. Indeed, a recently-designated precedential opinion clarifies that, together with the *General Plastic* factors, the Board also considers "any relationship between [the] petitioners" when "different petitioners challenge the same patent." *Valve*, IPR2019-00062, slip op. at 2.

Here, although this is the first petition by Petitioner or its indemnitee challenging the '969 Patent, it should be considered in the context of the other serially filed petitions on related patents and Petitioner's decision to inject itself in the district court litigation. Specifically, the relationship between Petitioner and its indemnitee, PayPal, the orchestrated effort by Petitioner to unnecessarily multiply these proceedings by first injecting itself into the district court forum—which has now consolidated both Petitioner's and IOENGINE's cases into a single, efficient, coordinated proceeding for all purposes until trial—and then, together with its indemnitee, filing ***three*** staged petitions on the '969 Patent, and ***twelve*** overall, and the overarching deficiencies in the Petition addressed in Section VII, weigh in favor of discretionary denial in their own right, and also cause most of the relevant factors to weigh in favor of discretionary denial.

1. **Factor 2: Petitioner Knew or Should Have Known of the Prior Art in this Petition at the Time of its Earlier Petitions**

Petitioner asserts in this Petition the same primary prior art reference, Iida, that it asserted in its three earlier petitions challenging the related '047 and '703 Patents. The earliest of those petitions was filed more than three months before this Petition. *See* 2072 at 5. Thus, Petitioner unquestionably knew of at least Iida sufficiently in advance of the time of filing that petition. In addition to Iida, this Petition also asserts the Fuji Guide and Shaffer. Petitioner asserts that the Fuji Guide was publicly available as early as 2001 and Shaffer is a patent published approximately two decades ago. Petitioner provides no explanation why it could not have found these references earlier through the exercise of reasonable diligence. *See Blue Coat Sys., Inc. v. Finjan, Inc.*, IPR2016-01441, slip op. at 12 (PTAB Jan. 23,

2017) (Paper 14); *see also Gen. Plastic*, IPR2016-01357, slip op. at 20; *Valve*, IPR2019-00062, slip op. at 10-11.

Here, Petitioner knew of Iida at least since it prepared and filed its earlier petitions, and should have known about the Fuji Guide and Shaffer, so the factor weighs against institution.

2. **Factors 4 & 5: Petitioner's Delay After Learning of the Prior Art Asserted in this Petition Weighs Against Institution**

More than three months elapsed between the time Petitioner unquestionably knew of Iida (and should have known of the Fuji Guide and Shaffer) and the time it filed the Petition, as discussed above. Petitioner provided no explanation for the delay. Under these circumstances, factors 4 and 5 weigh against institution. *See BASF SE v. Fresenius Med. Care Holdings, Inc.*, IPR2018-00283, slip op. at 9 (PTAB June 11, 2018) (Paper 7) (denying institution where petitioner knew of two references and should have known of the others at the time it filed the first petition, which was more than three months before filing the second petition).

3. **Factors 6, 7, & Stage of District Court Proceedings: Institution Would Be Highly Inefficient**

Institution in this matter would present a remarkably inefficient use of the Board's resources, leading to a race to judgment. The district court proceedings are coordinated with common dates until separate trials scheduled for July 27 and

August 10, 2020 for PayPal and Ingenico, respectively. Ex. 2031 ¶ 12. Because Petitioner chose to wait until just two weeks before the one-year statutory bar date based on service of the complaint on its indemnitee in the PayPal Action, a final written decision in this matter, if instituted, would be expected in mid-October 2020, which is ***after*** both district court trials. *Id.* The same is true for the remaining eight staged petitions filed by Ingenico and its indemnitee. Accordingly, the stage of the district court proceedings weighs in favor of denying the Petition. *See NHK Spring Co., Ltd. v. Intri-Plex Techs., Inc.*, IPR2018-00752, slip op. at 20 (PTAB Sep. 12, 2018) (Paper 8) (precedential).

In addition to a race to judgment, institution would also result in overlapping claim construction proceedings and ping-ponging decisions between the Board and the district court. The institution decision in this case would likely be in mid-October 2019, after claim construction briefing and the August 29 district court claim construction hearing are complete, and very close to the October 30 deadline for the close of fact discovery. Ex. 2031 ¶ 6. Institution decisions in the remaining eight cases would also be well after the August 29 claim construction hearing, and possibly after the October 30, 2019 close of fact discovery. *See* Ex. 2031 ¶¶ 3, 7.

Compounding the inefficiency of multiple, overlapping claim construction proceedings is the fact that the district court will likely rule on the very same claim construction issues presented by Ingenico here. Indeed, Ingenico has proposed both "interactive user interface" and "node" for construction in the district court with nearly identical constructions. *See* Ex. 2070.

Further, the validity issues raised by Petitioner and its indemnitee in the district court overlap those raised in the Petition. Ingenico has incorporated the Petition by reference into its invalidity contentions in the district court, along with all the other petitions that Ingenico and its indemnitee have filed. Ex. 2036 at 6-7. Moreover, Ingenico has presented additional theories involving Iida in the district court. *See* Ex. 2036 at 4. Ingenico has similarly raised Iida with respect to the '703 and '047 Patents. *See* Ex. 2036 at 4-5. Further, Ingenico has indicated it may present at trial any of the invalidity theories raised by PayPal in PayPal's invalidity contentions. Ex. 2036 at 7. PayPal in turn contends Iida anticipates the asserted claims of the '969 Patent. *See* Ex. 2040 at 9-10.

In view of the race to judgment, overlapping claim construction proceedings, and overlapping validity issues, factors 6, 7, and the stage of the district court proceedings weigh heavily against institution.

4. **Additional Factor: The Petition is Weak on the Merits**

As discussed in Section VII above, the Petition is weak on the merits. This factor further supports discretionary denial. *See* TPG Update at 25. Even if Petitioner has met its burden on a few claims, it has failed to meet its burden on so many claims and grounds that the entire petition should be denied. *See Chevron*

Oronite Co. LLC v. Infineum USA L.P., IPR2018-00923, slip op. at 10-11 (PTAB Nov. 7, 2018) (Paper 9) (informative).

IX. CONCLUSION

For the forgoing reasons, the Petition should be denied, and *inter partes* review should not be instituted.

Dated: *July 16, 2019*

/ Derek J. Brader /
Derek J. Brader, reg. no. 71,421
Robert W. Ashbrook, reg. no. 40,920
DECHERT LLP
Counsel for Patent Owner

WORD COUNT CERTIFICATION

Pursuant to 37 C.F.R. § 42.24(d), I certify that this Preliminary Response contains 13,397 words (excluding the title page, table of contents, table of exhibits, this certificate, and the certificate of service), as determined by Microsoft Word.

Dated: *July 16, 2019*

/ Derek J. Brader/
Derek J. Brader, reg. no. 71,421
DECHERT LLP
Counsel for Patent Owner

CERTIFICATE OF SERVICE

I certify that today in

Ingenico Inc. v. IOENGINE, LLC, IPR2019-00879

I caused to be served a copy of:

Patent Owner's Preliminary Response
Exhibits 2003-2004, 2018-2022, 2029-2033, 2035-2036, 2040, 2045-2046, 2062, 2069-2070, 2072-2076
Certificate of Service

upon:

Ingenico Inc., c/o:
Robert M. Asher, rasher@sunsteinlaw.com
Timothy M. Murphy, tmurphy@sunsteinlaw.com
Kerry L. Timbers, ktimbers@sunsteinlaw.com
sunsteinip@sunsteinlaw.com

via:

electronic service to the email addresses above.

Dated: *July 16, 2019*

/ Derek J. Brader /
Derek J. Brader, reg. no. 71,421
DECHERT LLP
Counsel for Patent Owner